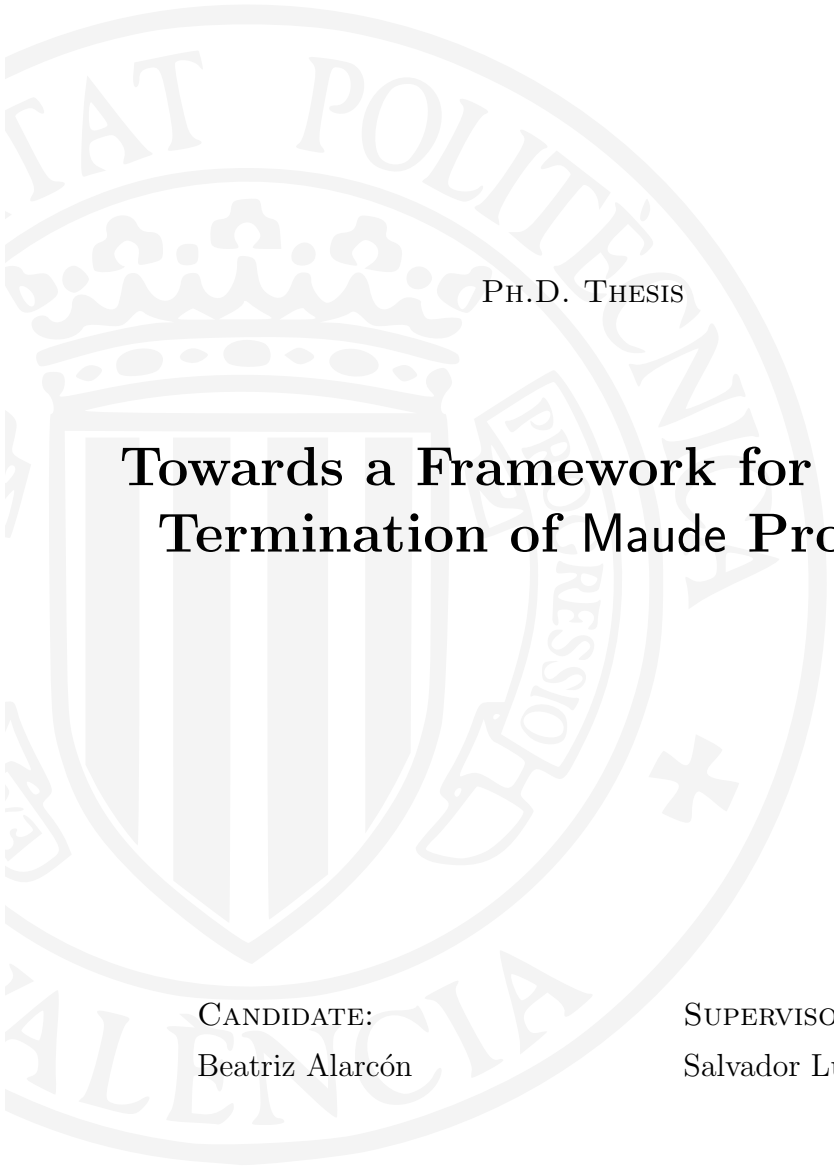
Universidad Politécnica de Valencia
Departamento de Sistemas Informáticos y Computación
Doctorado en Informática



Ph.D. Thesis

Towards a Framework for Proving Termination of Maude Programs

Candidate:
Beatriz Alarcón

Supervisor:
Salvador Lucas

– May 2011 –

This work has been partially supported by the EU (FEDER) and the Spanish MICINN, under grants TIN2010-21062-C02-02, TIN2007-68093-C02-02, TIN 2004-07943-C04-02 and HA 2006-2007, and the Generalitat Valenciana under grant GVPRE/2008/113. Also it was partially supported by the Spanish MICINN under FPU grant AP2005-3399.

Author's address:

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera, s/n
46022 Valencia
España

to Bienve, responsible for my happiness.

Acknowledgments

I would like to thank Salvador Lucas, my *scientific father*, for being a good advisor, not only for this thesis but also about life. He was often very strict, but just did what any good father would do. He has helped me appreciate the satisfaction of doing things well.

I would also like to thank María Alpuente for giving me the first opportunity to join the ELP family. It has been a pleasure to share these five years with all of them and the MIST group: Toni, Tama, Raúl, Rafa, Josep, Gustavo, Mauricio, Diego, Alexei, Sonia, Marco, Dani, Pepe, Nando, Alicia, Germán, Cesar, Santi, Jose, María José, ... and those that came only for a few months and gained a place in my heart: Vesna, Michele and Patricia.

Special mention goes to those that were not just workmates but also friends: Cesar (little bird) is the most incredible person I have met here, only he knows how much I owe him. Toni is the best lab mate ever. We have walked together from the beginning and we have shared many good moments. Tama, my spoiled child, I can only wish him all the best in the life, he certainly deserves it.

Many thanks to my coauthors, especially to Raúl, for all his help and support. I am proud of the great work that we have done together. It has been a pleasure having such a good companion along this difficult path. Thanks to Rafa for always having a *nice* word to say and making me laugh.

I would also like to thank all the people that I met around the world and allowed me to learn from them. These include Jürgen Giesl and the group at the RWTH Aachen: Carsten, Peter, Stephan, Ivan, René,...; and José Meseguer and the group at UIUC that made me feel at home: Camilo, Ralf, Edgar,... and specially Mike.

Special thanks go to my parents and brothers. I know you will never understand a word of this thesis, but you have always supported me in any decision I have made.

Finally, and the most importantly, my thanks to Bienve: my friend, my love, my life. Every dream is possible next to you.

May 2011,

Beatriz Alarcón

Abstract

Maude is a declarative programming language based on rewriting logic that incorporates many features that in order to prove certain computational properties lead to difficulties. The task of proving termination of rewrite systems in indeed quite hard but applied to real programming languages, becomes more complicated due to these inherent features. Therefore, methods for proving termination of such programs require specific techniques and a careful analysis. Several papers have studied how to prove termination of (a subset of) Maude programs. However, all of them follow a transformational approach where the original program is transformed until it reaches a rewrite system that can be managed with existing techniques and termination tools. In practice, the fact of transforming the original systems used to complicate the proof of termination since it introduces new symbols and rules in the system. In this thesis, we tackle the problem of proving termination of (a subset of) Maude programs by means of *direct* methods.

On the one hand, we pay attention to the strategy of Maude. Maude is an eager language where the arguments of a function are always evaluated before the application of the function that uses them. This strategy (known as *call by value*) can lead to nontermination if programs are not written carefully. For this reason, Maude (specifically) incorporates mechanisms to control the program execution such as syntactic annotations, which are associated to arguments of symbols. In rewriting, this strategy is known as *innermost context-sensitive rewriting*.

On the other hand, Maude also incorporates the possibility of declaring *attributes*. Semantically, declaring a set of equational attributes for an operator is equivalent to declaring the corresponding equations for the operator; however, it avoids termination problems and leads to a more efficient evaluation. The effect of declaring equational attributes is to compute with equivalence classes modulo these equations.

The *dependency pair framework* develops the idea of an incremental application of different termination techniques for solving termination problems. It has shown to be a powerful and efficient way to prove termination of rewriting automatically. In this thesis, we deal with termination of innermost context-sensitive rewriting and of rewriting modulo specific axioms by extending the dependency pair framework.

Resumen

Maude es un lenguaje de programación declarativo basado en la lógica de reescritura que incorpora muchas características que lo hacen muy potente. Sin embargo, a la hora de probar ciertas propiedades computacionales esto conlleva dificultades. La tarea de probar la terminación de sistemas de reesctritura es de hecho bastante dura, pero aplicada a lenguajes de programación reales se concierte en más complicada debido a estas características inherentes. Esto provoca que métodos para probar la terminación de este tipo de programas requieran técnicas específicas y un análisis cuidadoso. Varios trabajos han intentado probar terminación de (un subconjunto de) programas Maude. Sin embargo, todos ellos siguen una aproximación transformacional, donde el programa original es trasformado hasta alcanzar un sistema de reescritura capaz de ser manejado con las técnicas y herramientas de terminación existentes. En la práctica, el hecho de transformar los sistemas originales suele complicar la demostración de la terminación ya que esto introduce nuevos símbolos y reglas en el sistema. En esta tesis, llevamos a cabo el problema de probar terminación de (un subconjunto de) programas Maude mediante métodos directos.

Por un lado, nos centramos en la estrategia de Maude. Maude es un lenguaje impaciente donde los argumentos de una función son evaluados siempre antes de la aplicación de la función que los usa. Esta estrategia (conocida como *llamada por valor*) puede provocar la no terminación si los programas no están escritos cuidadosamente. Por esta razón, Maude (en concreto) incorpora mecanismos para controlar la ejecución de programas como las anotaciones sintácticas que están asociadas a los argumentos de los símbolos. En reescritura, esta estrategia sería conocida como *reescritura sensible al contexto innermost (RSCI)*.

Por otro lado, Maude también incorpora la posibilidad de declarar *atributos*. Semánticamente, declarar un conjunto de atributos ecuacionales para un operador es equivalente a declarar las ecuaciones correspondientes para el operador pero evita problemas de terminación y provoca una evaluación más eficiente. Declarar atributos ecuacionales se corresponde con computar con clases de equivalencia módulo esas ecuaciones.

El marco de los pares de dependencia desarrolla la idea de una aplicación incremental de diferentes técnicas de terminación para resolver problemas de terminación. Se ha mostrado como una manera potente y eficiente de probar

terminación de la reescritura automáticamente. En esta tesis, abordamos la terminación de la reescritura sensible al contexto innermost y de la reescritura módulo axiomas específicos extendiendo el marco de los pares de dependencia.

Resum

Maude és un llenguatge de programació declaratiu basat en lògica de reescriptura que incorpora moltes característiques que ho fan molt potent. No obstant això, a l'hora de provar certes propietats computacionals este factor comporta dificultats. La tasca de provar la terminació de sistemes de reescriptura és de fet bastant dura, però aplicada a llenguatges de programació reals es convert en més complicada a causa de aquestes característiques inherents. Açò provoca que mètodes per a provar la terminació d'aquest tipus de programes requerisquen tècniques específiques i una anàlisi exhaustiu. Diversos treballs han intentat provar terminació de (un subconjunt de) programes Maude. No obstant això, tots ells segueixen una aproximació transformacional, on el programa original és transformat fins a arribar a un sistema de reescriptura capaç de ser empleat amb les tècniques i eines de terminació existents. En la pràctica, el fet de transformar els sistemes originals sol complicar la demostració de la terminació ja que açò introdueix nous símbols i regles en el sistema. En aquesta tesi, abordem el problema de provar terminació de (un subconjunt de) programes Maude mitjançant mètodes directes.

D'una banda, ens centrem en l'estratègia de Maude. Maude és un llenguatge impacient on els arguments d'una funció són avaluats sempre abans de l'aplicació de la funció que els usa. Aquesta estratègia (coneguda com *crida per valor*) pot provocar la no terminació si els programes no estan escrits amb cura. Per aquesta raó, Maude (en concret) incorpora mecanismes per controlar l'execució de programes com les anotacions sintàctiques que estan associades als arguments dels símbols. En reescriptura, aquesta estratègia es coneix com *reescriptura sensible al context innermost (RSCI)*.

D'altra banda, Maude també incorpora la possibilitat de declarar *atributs*. Semànticament, declarar un conjunt d'atributs equacionals per a un operador és equivalent a declarar les equacions corresponents per a l'operador però evita problemes de terminació i provoca una avaluació mes eficient. Declarar atributs equacionals es correspon amb computar amb classes d'equivalència mòdul aquestes equacions.

El marc dels parells de dependència desenvolupa la idea d'una aplicació incremental de diferents tècniques de terminació per a resoldre problemes de terminació. S'ha mostrat com una manera potent i eficient de provar terminació de la reescriptura automàticament. En aquesta tesi, abordem la terminació

de la reescriptura sensible al context innermost i de la reescriptura mòdul axiomes específics estenent el marc dels parells de dependència.

Contents

List of Figures

List of Tables

1

Introduction

Termination is a fundamental property that programs in software systems must often satisfy and is also a prerequisite for demonstrating other properties in program verification. A program is terminating if it does not lead to infinite computations for any given input data. In the last few years, many studies have been developed to analyze termination of programming languages, mainly of functional [Gie95, LJB01, Xi02] and logic programming languages [CLS05, CT99, DD94, DLSS01, DS02, LMS03, Sma04]. In the case of imperative programming languages, it is becoming important as well [AAC+08, BMS05, CPR06, CS02, Tiw04]. Since most computational systems whose operational principle is based on reducing expressions can be described and analyzed by using notions and techniques that come from the abstract model of Term Rewriting Systems (TRSs [BN98, TeR03]), in many programming languages like Haskell, Maude, etc., it is possible to reduce the proofs of termination of the corresponding programs to proofs of termination of (variants of) TRSs. For this reason, the development of techniques for proving termination of term rewriting systems becomes especially important since every improvement will have a positive impact on program verification for many programming languages. Following this approach, a number of results, techniques, and implementations have been developed for the aforementioned programming languages. With regard to termination of logic programs, several works can be found: [AM93, KKS98, Mar94, Mar96, SGN09, SGST06]. Termination of the functional language Haskell [HPW92] has been investigated quite recently [GRS+10] as well as termination of Java Bytecode [OBEG10].

A *Term Rewriting System* (TRS) is a pair $\mathcal{R} = (\Sigma, R)$, where R is a set of rewrite rules and Σ corresponds to a signature. As usual, by a *signature*, we mean a set of function symbols $f_1, f_2, \ldots$ together with an *arity* function $\mathsf{ar} : \Sigma \to \mathbb{N}$ which establishes the number of 'arguments' associated to each function symbol. A *rewrite rule* is an ordered pair (l, r), written $l \to r$, where l and r are *terms* such that l is not a variable, and variables occurring in r

also occur in l. In general, the termination of term rewriting systems is an undecidable property [HL78]: Since it is undecidable (not even semi-decidable) whether a Turing machine halts uniformly, and Turing machines can be simulated by rewriting systems, termination of rewrite systems is also undecidable (even for linear one-rule systems [Dau92]). Many techniques have been developed so far to try to analyze termination of a wide range of systems and, obviously, there is still major interest if these techniques are fully mechanizable. At the beginning, these techniques focused on finding *reduction orders* on terms, that is, (strict) partial orders which are monotonic (closed under context), stable (closed under substitution) and with no infinite decreasing sequences [Lan77, Lan79, Der87]. If we find a reduction order such that, for all rules of the TRS, the left-hand sides are greater than the corresponding right-hand sides, then the TRS is terminating. There are two main methods for generating reduction orders on terms [BN98, Ohl02]: The interpretation method and simplification orders . The interpretation method [MN70] does not look directly at the terms over the signature Σ. Instead, it considers their interpretation in a Σ-algebra under a well-founded order. Polynomial interpretations are special well-founded algebras in which function symbols are interpreted as polynomials. They were first studied by Lankford [Lan75, Lan79]. It is also possible to define rewrite orders (monotonic and stable orders) directly on terms by using simplification orders [Der79], i.e., rewrite orders that satisfy the subterm property[1]. Simplification orders represented the basis for automatic proofs of termination of rewriting. They can be classified in three ways: syntactic, like the lexicographic path order (LPO [KL80]) or the recursive path order (RPO [Der82]); semantic, like the polynomial simplification orders [Lan79]; and the ones combining both characteristics like the Knuth-Bendix order (KBO [KB70]).

Example 1

Consider the following TRS $\mathcal{R}$ for integer division of natural numbers [AG00]:

$$\begin{aligned}
\mathtt{minus}(x, \mathtt{0}) &\to x \\
\mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\to \mathtt{minus}(x, y) \\
\mathtt{quot}(\mathtt{0}, \mathtt{s}(y)) &\to \mathtt{0} \\
\mathtt{quot}(\mathtt{s}(x), \mathtt{s}(y)) &\to \mathtt{s}(\mathtt{quot}(\mathtt{minus}(x, y), \mathtt{s}(y)))
\end{aligned}$$

This TRS is not compatible with a simplification order because the left-hand side of the last rule of `quot` is embedded in its right-hand side if y is instantiated with $\mathtt{s}(x)$. Therefore, these techniques cannot prove termination of this TRS.

[1] For all terms s, and strict subterms t of s, we have $s > t$.

In [AG00], a new technique emerged to solve this problem: the *dependency pair* (DP) approach. The central idea of this approach is to compare left-hand sides of rules only with those subterms of the right-hand sides that may possibly start a new reduction. The motivation for this approach is to regard TRSs as 'programs'. Intuitively, such a program is terminating if the arguments are decreasing in each recursive call. Then, numerous term rewriting systems where a direct termination proof with simplification orders was not possible can now use existing simplification orders to prove termination automatically in combination with the dependency pairs. During the last decade, the dependency pair method has evolved into a powerful technique for proving termination of TRSs in practice. Apart from the DP approach, there also exist quite powerful techniques for proving termination of term rewriting systems like *semantic labeling* [Zan95], the monotonic semantic path order [BFR00], and *match-bounds* [GHW03, GHWZ07] that were seen as independent techniques before the latest developments corresponding to the so-called *dependency pair framework* (DP framework [GTS04, GTSF06, Thi07]). From the already classical Arts and Giesl article [AG00] to now, many new improvements have been introduced (see [GAO02, GTS04, GTS05, HM04, HM05] for refinements and motivations); now, even these "independent" techniques can be used inside the DP framework (see for instance [KM09]). Although the DP approach emphasizes a 'linear' procedure for proving termination (at least theoretically), the DP framework leads to a more powerful mechanization of termination proofs in an incremental and modular way.

In comparison to direct methods for proving termination of TRSs, several approaches have been developed to deal with termination of (variants of) rewriting by means of transformations. This is the case of context-sensitive rewriting [Luc96, Zan97, FR99, GM99], innermost context-sensitive rewriting [GM02a, Luc01a], and order-sorted rewriting [ÖL96, DLM+08, LM09] among others. However, this approach has been proved to be less powerful than applying intrinsic methods since it introduces many new symbols and rules to the original systems, making the proof of termination difficult.

In recent years, we have worked to extend the DP framework to verify a number of termination properties of (variants of) TRSs: termination of context-sensitive rewriting [AGL06, AGL07, AGL10], termination of innermost context-sensitive rewriting [AL07, AL09], termination of order-sorted rewriting [LM08], and termination of rewriting modulo specific axioms [ALM10, AGLM11]. The main reason is because many systems can be nonterminating if those features are not taken into account when proving termination like

strategy annotations or sorts. Also, other systems can wrongly be proved terminating if some features are not considered like specific axioms associated to function symbols (eg., associativity, commutativity, etc.).

These features are essential in many programming languages and in Maude in particular.

1.1 Maude

Maude [CDEL+07] is an executable specification language that is also considered as a programming language whose precursor is the OBJ3 language [GWM+00].

Maude is based on rewriting logic [BM03], and its modules are rewrite theories. Since Maude is based on logic and has an initial model semantics, a Maude module has a precise mathematical model. This brings the possibility of using Maude in three ways: as a declarative programming language; as an executable formal specification language; and as a formal verification system. Even though Maude's rewriting logic is quite simple, it is very expressive and provides good capabilities as a *semantic framework* [CDEL+07] to formally represent a wide range of systems. Furthermore, rewriting logic is also an expressive universal logic. Therefore, many different logics and inference systems can be represented under its *logical framework* [CDEL+07]. Three are the dimensions that Maude tries to maximize: simplicity, expressiveness, and performance. However, the key point in Maude's language design is to maximize expressiveness. Maude can express both *deterministic* computations (which lead to a unique final result) and *concurrent, nondeterministic* computations with equal ease. The first kind refers typically to *functional* modules in Maude whereas the second one is handled with *system* modules. In fact, functional modules define a functional sublanguage of Maude, which is essentially an extension of OBJ3. On the other hand, system modules extend the purely functional semantics of equations (in functional modules) to the concurrent rewriting semantics of rules. Apart from being able to express both deterministic and nondeterministic computations, further expressiveness is gained by: equational pattern matching; user-definable syntax and data; types, subtypes, and partiality; generic types and modules; support for objects; and reflection.

Maude is a declarative language, that is, a Maude program is a logical theory and a Maude computation is logical deduction using the axioms specified in the theory/program. Since functional modules can be seen as a special case of system modules, at the mathematical level, this inclusion is precisely the sublogic inclusion in which *membership equational logic* (MEL

[Mes98, BJM00]) (an extension of order-sorted equational logic [GM92]) is embedded in *rewriting logic* (RL) [Mes92, BM03]. A functional module specifies a *theory* in MEL. Mathematically, we can view such a theory as a pair $(\Sigma, E \cup A)$. Σ is the signature, which specifies the type structure: sorts, subsorts, kinds, and (overloaded) operators. E is the collection of (possibly conditional) equations and memberships declared in the functional module, and A is the collection of equational attributes (associativity, commutativity, etc.) that is declared for the different operators. In relation to computation, it is assumed that equations have been given in such a way that they can be efficiently executed by applying them from left to right. This process is called equational rewriting or equational simplification because, intuitively, the expressions get progressively simpler. This is of course a special form of equational deduction in which equations are used from left to right as simplification rules. Similarly, a system module specifies a *rewrite theory*, that is, a theory in rewriting logic. Mathematically, such a rewrite theory is a 4-tuple $\mathcal{R} = (\Sigma, E \cup A, \phi, R)$, where $(\Sigma, E \cup A)$ is the module's equational part, ϕ is the function specifying the frozen arguments of each operator in Σ, and R is a collection of (possibly conditional) rewrite rules. Computation is rewriting logic deduction, in which equational simplification with the axioms $E \cup A$ is intermixed with rewriting computation with the rules R. Apart from the obvious inclusion of functional modules into the class of system modules (where there are no rules and no argument is frozen), Maude also allows the user to give the desired freezing information for each operator in the signature of a functional module. Each Maude module not only specifies a theory, but also an intended mathematical model, the one that the user has intuitively in mind. For functional modules, these models consist of sets of data and functions defined on this data known as *algebras*. Mathematically, the intended model of a functional module that specifies an equational theory $(\Sigma, E \cup A)$, with Σ the signature defining the sorts, subsorts, and operators, E the equations and memberships, and A the equational attributes like `assoc`, `comm`, and so on is called the *initial algebra* of such a theory and is denoted $T_{\Sigma/E \cup A}$. Similarly, a system module that specifies a theory $\mathcal{R} = (\Sigma, E \cup A, \phi, R)$ has an initial model, which in essence is an algebraic *(labeled) transition system*. The states and data of the system are elements of the underlying initial algebra $T_{\Sigma/E \cup A}$. The state transitions are the concurrent rewrites by application of the rules R. Both kinds of models associated with Maude modules fit together with the computations under the so-called agreement between the *mathematical semantics* (the models) and the *operational semantics* (the computations). The key idea is that under certain executability conditions required of Maude modules, both semantics coincide. In the case of functional modules (and in

the equational part of system modules), the equations, considered as simplification rules, have to be Church-Rosser[2] and terminating. This means that repeated application of the equations eventually reaches a term to which no further equations apply, and the result called the *canonical form* (or more commonly used in rewriting, *normal form*), which is the same regardless of the order of application of the equations. Thus, each equivalence class has a natural representative, its normal form. However, even though the final result may be the same, some order of evaluation may be considerably more efficient than others. In fact, we may lose termination when any evaluation order is allowed. Therefore, it may be useful to have some way of controlling the order in which equations are applied by means of strategies. Typically, a functional language is either eager or lazy and the user has to live with whatever the language provides. Maude adopts the OBJ3 flexible method of user-specified *evaluation strategies* on an operator. For an n-ary operator f, an evaluation strategy is specified as a list of numbers from 0 to n ending with 0. The nonzero numbers denote argument positions, and a 0 indicates evaluation at the top of the given function symbols. Then, the strategy specifies what argument positions must be simplified before attempting simplification at the top with the equations and also in which argument positions simplification is not allowed (the missing argument positions in the strategy list). In Maude, if no strategy is specified, for an operator f with n argument positions, its default strategy is $(1\ 2 \ldots n\ 0)$, that is, the *eager* evaluation case, what in rewriting strategies is known as *innermost* evaluation. The syntax to declare an n-ary operator with strategy $(i_1 \ldots i_k\ 0)$, where $i_j \in \{0, \ldots, n\}$ for $j = 1, \ldots, k$ is

`op` ⟨*OpName*⟩ `:` ⟨*Sort-1*⟩ ... ⟨*Sort-n*⟩ `->` ⟨*Sort*⟩ `[strat (i1 ... ik 0)]`

For example, an `if_then_else_fi` operator will typically be evaluated by first evaluating the first argument, and then the `if_then_else_fi` operator at the top.

Therefore, equational specifications in Maude are assumed to be Church-Rosser and terminating up to the context-sensitive strategy specified by the evaluation strategies declared for the operators in Σ. More precisely, the information about the evaluation strategy of each operator can be seen as the so-called replacement map μ in context-sensitive rewriting (*CSR*) [Luc98, Luc02]. Thus, instead of termination, we have to talk about *μ-termination* as a requirement for achieving the coincidence of both semantics in Maude programs. This coincidence is crucial for reasoning about Maude programs and verifying their correctness.

[2] *A* binary reduction relation $\rightarrow \subseteq \mathsf{A} \times \mathsf{A}$ on a set A is Church-Rosser if, for all $a, b \in \mathsf{A}$ with $a \leftrightarrow^* b$, the elements a and b have a common reduct c, i.e., $a \rightarrow^* c$ and $b \rightarrow^* c$.

Example 2

Consider the following Maude program which defines the addition of natural numbers in a functional module with Peano notation, so that zero is represented as the constant `0`, and there is a successor function `s_`.

```
fmod PEANO-NAT is
sort Nat .
op 0 : -> Nat .
op s : Nat -> Nat .
op plus : Nat Nat -> Nat .
vars N M : Nat .
eq plus(0, M) = M .
eq plus(s(N), M) = s(plus(N, M)) .
endfm
```

In this example, we have the following from the signature of `PEANO-NAT`: one sort, `Nat`, and three operators, `0`, `s`, and `plus`. Sorts are declared with the keyword `sort`, and operators with the keyword `op`. The three operators have zero, one, and two arguments, resp., whose sorts are between `:` and `->`. Operators of zero arguments are also called constants; those of one argument are called unary, and those of two binary. The result sort appears after `->`.

The two equations are *properties* that these operators should satisfy. More precisely, any correct implementation of Peano natural numbers should satisfy them.

There can exist different classes of signatures in Maude:

- Unsorted (or single-sorted) signatures have only one sort and operation symbols defined on it.
- Many-sorted signatures allow different sorts, such as `Int`, `Bool`, `List`, etc. and operations defined on them.
- Order-sorted signatures are many-sorted signatures that, in addition, allow inclusion relations between sorts, such as `Natural < Integer`.

In general, the same operator *name* may have different declarations in the same signature Σ. For example, Maude allows subsort overloading like:

```
op plus  :  Nat Nat -> Nat.
op plus  :  NzNat NzNat -> NzNat.
```

having `Nznat < Nat` and also ad-hoc overloading like:

```
op plus  :  Nat Nat -> Nat.
op plus  :  List List -> List .
```

where the sorts are not related in the signature Σ.

Example 3

Consider two relevant properties of natural numbers addition, namely, associativity and commutativity. These properties are described by the following equations

```
eq plus(N, M) = plus(M, N) .
```

```
eq plus(N, plus(M, L)) = plus(plus(N, M), L) .
```

It is easy to see that these equations are not provable by equational deduction, that is, they do not follow by replacing equals by equals from the two equations that define the addition function. These equations should not be written explicitly as equations in the specification since declaring such equations drastically alters the specification's operational semantics. For example, if the commutative equation is used as a simplification rule to the term, say, `plus(s(0),s(s(0)))`, it would lead to the nonterminating sequence of equational simplification

```
plus(s(0),s(s(0))) = plus(s(s(0)),s(0)) = plus(s(0),s(s(0))) = ...
```

Maude solves this problem by using *equational attributes*, thus avoiding the previous loop. Then, Maude simplifies terms modulo the declared equational attributes, so that the terms `plus(s(0),s(s(0)))` and `plus(s(s(0)),s(0))` would be treated as identical.

Therefore, the way of dealing with operators that satisfy certain properties like associativity and commutative should be by using Maude's definition of equational attributes. In particular, for the addition program, it would have to be declared by adding

```
op plus :  Nat Nat -> Nat [assoc comm] .
```

Semantically, declaring a set of equational attributes for an operator is equivalent to declaring the corresponding equations for the operator. Operationally,

using equational attributes to declare these equations avoids termination problems and leads to a more efficient evaluation. The effect of declaring equational attributes is to compute with equivalence classes modulo such equations. Note that, when equational attributes are declared, equational simplification using the other equations in the module does not take place at the purely syntactic level of replacing equals by equals, but it is understood *modulo* the declared equational attributes. Therefore, the proper understanding of the notions of Church-Rosser and termination, and of normal forms have to be considered modulo the equational attributes.

More information about the wide variety of features that Maude has, its syntax and everything related to it can be found in [CDEL+07]. Our aim is to show the importance of proving termination of Maude programs by emphasizing the features that we have highlighted in this work.

Obviously, since Maude is a general-purpose declarative programming language, in principle there is no limit to the applications that could be developed using it. To enumerate some diverse application areas of rewriting logic and Maude (see [CDEL+07] for a more complete list of references):

1. Models of computation like: equational programming, lambda calculi, labeled transition systems, grammars, Petri nets, π calculus, dataflow, neural networks, etc.
2. Semantics of programming languages and software analysis.
3. Maude as a Metalanguage.
4. Modeling and analysis of networks and distributed systems.
5. Real-Time Systems.
6. Probabilistic Systems.
7. Modeling and analysis of biological systems.

For all these reasons, proving termination of Maude programs is an important topic that has been studied in recent years, mainly, by means of transformations as we explain in the next section.

1.2 Proving Termination of Maude Programs by Transformation

Despite the huge development of the theory of termination of rewriting, its application to high-level rewriting-based programming languages is quite im-

mature. In rewriting-based programming languages such as CafeOBJ, ELAN, or Maude, one is often tempted to map termination problems for programs in such languages directly into termination problems for TRSs or conditional TRSs (CTRSs, see [Ohl02] for results in this field) in a quite straightforward way. However, handling programs in this way can often lead to wrong conclusions about their real termination behavior. The main reason is that these programs make use of additional features whose appropriate consideration is often essential to prove termination, but which are not captured by the computational model of pure term rewriting: sorts, subsorts, and operator overloading; memberships, conditions, evaluation strategies, rewriting modulo axioms, etc. Over the last few years, different transformation techniques have been introduced, implemented, and proved useful in proving termination of these programs (see [DLM09a] for a survey). In [DLM09a] (and in the papers surveyed there), the termination problem for rewrite theories is investigated. A number of theory transformations Θ that have been developed are informally described. These transformations are nontermination preserving. Thus, a rewrite theory can be mapped into a transformed TRS that can be proved terminating by using standard termination tools. The different kind of logics/theories/programs that are transformed are:

- *(Sugared) Rewrite theories (RWT).* A rewriting logic specification is called a rewrite theory. It is a tuple $\mathcal{R} = (\Sigma, E \cup A, \mu, R, \phi)$ where each element has the same meaning as the homonym stated in the previous section.

- *Sugared Context-Sensitive Membership Rewrite Theories (SCS-MCTRSs).* By sugared context-sensitive membership rewrite theories [LM09], we understand a tuple $\mathcal{R} = (\Sigma, S, \leq, \mu, A, R, M)$ with new elements:

  - S is a set of sorts and $(S, \leq)$ is a partial order.
  - M is a set of conditional memberships. Membership axioms specify terms as having a given sort.

    `cmb` ⟨*Term*⟩ : ⟨*Sort*⟩
    `if` ⟨*EqCondition-1*⟩ `/\` ... `/\`⟨*EqCondition-k*⟩
    `[`⟨*StatementAttributes*⟩`]` .

- *Order-Sorted Rewrite Theories (OS-RWT).* It consists of a tuple $\mathcal{R} = (\Sigma, S, \leq, E \cup A, \mu, R, \phi)$ where each element has the same meaning as the homonym in previous definitions [DLM09a].

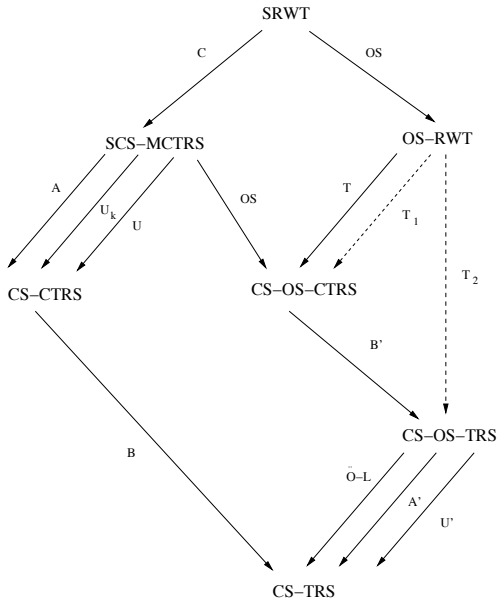


Figure 1.1: Transformations for proving termination of Maude programs

- *Conditional and Context-Sensitive TRSs (CTRS/CS-TRS/CS-CTRS).* A conditional TRS (CTRS) is a triple $\mathcal{R} = (\Sigma, A, R)$. A context-sensitive CTRS (CS-CTRS) is a pair $(\mathcal{R}, \mu)$ with $\mathcal{R}$ being a conditional TRS (CTRS) and μ a replacement map that satisfies $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f in the signature Σ [Luc98, Luc02]. The replacement map μ discriminates the argument positions $\mu(f)$ of function symbols f where rewritings are allowed in the context-sensitive TRS (CS-TRS) $(\mathcal{R}, \mu)$.

The overall family of composable nontermination-preserving transformations is summarized in Fig. 1.1 (see [DLM09a] for details). We now briefly describe how some of these transformations proceed.

- *Transformation C* : from SRWTs to SCS-MCTRSs [DLM08b]. This simple transformation merges equations E and rules R.

- *Transformation A* : from SCS-MCTRSs/CS-OS-CTRSs to CS-CTRSs [DLM+08]. This transformation allows us to deal with sort information, subsort declarations, rank declarations for symbols in the signature, sorted variables, etc. *Transformations U_k and U* were also discussed

in [DLM+08] as increasingly simpler lightweight variants of A: U_k ignores kind information, but still encodes sort information as predicates; whereas U ignores both kind and sort information.

- *Transformation OS* : from SCS-MCTRSs to CS-OS-CTRSs. The transformation OS is described in detail in [LM09] and maps an SCS-MCTRS to a CS-OS-CTRS dealing explicitly with memberships. It can also be used from SRWTs to OS-RWTs.

- *Transformation T* : from OS-RWTs to CS-OS-CTRSs. This transformation deals with encoding equational rewriting, providing only a single rewrite relation. If the rewrite theory $\mathcal{R}$ satisfies some conditions, this transformation can be even simpler; in these cases they are denoted T_1 and T_2. T_2 has the advantage that the transformed theory is always an unconditional OS-TRS.

- *Transformation B* : from CS-CTRSs to CS-TRSs [DLM+08]. It generalizes, to the CS-case, a well-known transformation from CTRSs to TRSs described in [Ohl02].

- *Transformation B′* : from CS-OS-CTRSs to CS-OS-TRSs [LM09]. This transformation plays a similar role than B for the order-sorted case.

- *Transformation Ö-L* : from CS-OS-TRSs to CS-TRSs [LM09]. It generalizes, to the context-sensitive level, a well-known transformation by Ölveczky and Lysne [ÖL96].

The experiments made with these transformations suggest that they can be effective in proving termination of a wide range of rewriting logic programs. However, the authors believe [DLM09a] that these techniques should be combined with more *intrinsic* techniques to keep some information around and use it directly in termination proofs rather than encoding it into new conditions and rules that, sometimes, make the termination proof harder. For instance, the following specification of the factorial function [LM08]:

```
fmod FACTORIAL is
  sorts Nat NzNat .
  subsorts NzNat < Nat .
  op 0 : -> Nat .
  op s : Nat -> NzNat .
  op p : NzNat -> Nat .
  op _+_ : Nat Nat -> Nat .
```

```
  op _+_ : NzNat Nat -> NzNat .
  op _+_ : NzNat NzNat -> NzNat .
  op _*_ : Nat Nat -> Nat .
  op _*_ : NzNat NzNat -> NzNat .
  op fact : Nat -> NzNat .
  vars X Y : Nat .
  vars X' : NzNat .
  eq X + 0 = X .
  eq X + s(Y) = s(X + Y) .
  eq X * 0 = 0 .
  eq X * s(Y) = X + (X * Y) .
  eq fact(0) = s(0) .
  eq fact(X') = X' * fact(p(X')) .
  eq p(s(X)) = X .
endfm
```

can be easily proved terminating as an OS-TRS by using the recently introduced order-sorted dependency pair (OS-DP) method [LM08], which is implemented in the termination tool MU-TERM [AGLN10, AGIL07, Luc04]. However, it is not possible to obtain an automatic proof of termination using the existing transformations described in this section which are implemented in the Maude Termination Tool: MTT[3] [DLM08a]. Therefore, there is an important need to develop direct methods for proving termination of Maude programs at the different theory levels depicted in Fig. 1.1.

1.3 Dependency Pairs and the DP framework

A TRS $\mathcal{R}$ is terminating if there is no infinite rewrite sequence starting from any term. With regard to proofs of termination of rewriting, the dependency pair technique focuses on the following idea: the rules that are really able to produce such infinite sequences are those rules $l \to r$ such that r contains some *defined symbol*[4] g. Intuitively, we can think of these rules as representing some possible (direct or indirect) recursive calls. Such recursion paths associated to each rule $l \to r$ are represented as new rules $u \to v$, where $u = f^\sharp(l_1, \ldots, l_k)$ if $l = f(l_1, \ldots, l_k)$, and where $v = g^\sharp(s_1, \ldots, s_m)$ if $s = g(s_1, \ldots, s_m)$ is a subterm of r and g is a defined symbol. The notation $f^\sharp$ for a given symbol f means that f is *marked*. In practice, we often capitalize f and use F instead

[3] http://www.lcc.uma.es/~duran/MTT

[4] A symbol $g \in \Sigma$ is defined in $\mathcal{R}$ if there is a rule in $\mathcal{R}$ whose left-hand side is of the form $g(l_1, \ldots, l_k)$.

of $f^\sharp$ in our examples. For this reason, the dependency pair technique starts by considering a new TRS $\mathsf{DP}(\mathcal{R})$ that contains all these new rules for each $l \to r \in \mathcal{R}$. The rules in $\mathcal{R}$ and the rules in $\mathsf{DP}(\mathcal{R})$ determine the so-called *dependency chains* whose finiteness characterize termination of $\mathcal{R}$ [AG00]. A *chain of dependency pairs* is a sequence $u_i \to v_i$ of dependency pairs together with a substitution σ such that $\sigma(v_i)$ rewrites to $\sigma(u_{i+1})$ for all $i \geq 1$.

Theorem 4 [AG00] *A TRS $\mathcal{R}$ is terminating iff there exists a weakly monotonic quasi-order $\geq$, where both $\geq$ and its strict part $>$ are closed under substitutions, such that $>$ is well-founded and*

- *$l \geq r$ for all rules $l \to r \in \mathcal{R}$ and*
- *$u > v$ for all rules $u \to v \in \mathsf{DP}(\mathcal{R})$.*

In the DP approach, well-foundedness is not required for the quasi-order $\geq$ that is used to compare the rules. Furthermore, monotonicity is *not* required for the strict and well-founded order $>$ that is used to compare the DPs. This permits a more flexible use of orders.

Example 5

Continuing with Example 1, we have the following set of dependency pairs $\mathsf{DP}(\mathcal{R})$:

$$\begin{array}{rcl} \mathtt{MINUS}(\mathtt{s}(x),\mathtt{s}(y)) & \to & \mathtt{MINUS}(x,y) \\ \mathtt{QUOT}(\mathtt{s}(x),\mathtt{s}(y)) & \to & \mathtt{QUOT}(\mathtt{minus}(x,y),\mathtt{s}(y)) \\ \mathtt{QUOT}(\mathtt{s}(x),\mathtt{s}(y)) & \to & \mathtt{MINUS}(x,y) \end{array}$$

By means of the following polynomial interpretation, each function symbol is interpreted as a polynomial over the natural numbers:

$$\begin{array}{rclrclrcl} [\mathtt{0}] & = & 0 & [\mathtt{s}](x) & = & 2x+2 & [\mathtt{minus}](x,y) & = & x \\ [\mathtt{quot}](x,y) & = & x & [\mathtt{QUOT}](x,y) & = & x+y & [\mathtt{MINUS}](x,y) & = & 2x+2y \end{array}$$

Then, we have the following:

$$\begin{array}{rclcrcl} [\mathtt{minus}(x,\mathtt{0})] & = & x & \geq & x & = & [x] \\ [\mathtt{minus}(\mathtt{s}(x),\mathtt{s}(y))] & = & 2x+2 & \geq & x & = & [\mathtt{minus}(x,y)] \\ [\mathtt{quot}(\mathtt{0},\mathtt{s}(y))] & = & 0 & \geq & 0 & = & [\mathtt{0}] \\ [\mathtt{quot}(\mathtt{s}(x),\mathtt{s}(y))] & = & 2x+2 & \geq & 2x+2 & = & [\mathtt{s}(\mathtt{quot}(\mathtt{minus}(x,y),\mathtt{s}(y)))] \\ [\mathtt{MINUS}(\mathtt{s}(x),\mathtt{s}(y))] & = & 4x+4y+8 & > & 2x+2y & = & [\mathtt{MINUS}(x,y)] \\ [\mathtt{QUOT}(\mathtt{s}(x),\mathtt{s}(y))] & = & 2x+2y+4 & > & x+2y+2 & = & [\mathtt{QUOT}(\mathtt{minus}(x,y),\mathtt{s}(y))] \\ [\mathtt{QUOT}(\mathtt{s}(x),\mathtt{s}(y))] & = & 2x+2y+4 & > & 2x+2y & = & [\mathtt{MINUS}(x,y)] \end{array}$$

Therefore, the requirements in Theorem 4 are fulfilled and termination of $\mathcal{R}$ can be proved.

The DPs can be presented in a *dependency graph*, where the *infinite chains* are represented by the *cycles* in the graph. In this dependency graph, we can decompose the problem of proving termination of a TRS into the problem of proving the absence of infinite chains of DPs that are part of the cycles in the graph. This modular decomposition permits the use of different orders with different cycles [GAO02].

As we have stated, the DP approach emphasizes (at least theoretically) a 'linear' procedure for proving termination. In the DP approach, dependency pairs are considered as components of the chains (or cycles). Since they only make sense when an underlying TRS is given as the source of the dependency pairs, transforming DPs is possible (the *narrowing* transformation is already described in [AG00]) but only as a final step because, afterwards, they are no longer dependency pairs of the original TRS. The *DP framework* solves these problems in a clear way, leading to a more powerful mechanization of termination proofs. The crucial feature of the DP framework when dealing with proofs of termination is to examine a set $\mathcal{P}$ of pairs, which are intended to be (possibly transformed) DPs, together with the rules $\mathcal{R}$ to prove the absence of (minimal) infinite $(\mathcal{P},\mathcal{R})$-chains. A $(\mathcal{P},\mathcal{R})$-chain is a sequence $u_1 \to v_1, u_2 \to v_2, \ldots$ of pairs $u_i \to v_i \in \mathcal{P}$ together with a substitution σ such that $\sigma(v_i)$ rewrites to $\sigma(u_{i+1})$ for all $i \geq 1$. In the DP framework, the central notion regarding termination proofs is that of *DP problem* : given a TRS $\mathcal{R}$ and a set of pairs $\mathcal{P}$, the goal is to check the absence (or presence) of infinite (minimal) chains. Termination of a TRS $\mathcal{R}$ is addressed as a DP problem where $\mathcal{P} = \mathsf{DP}(\mathcal{R})$. The most important notion regarding mechanization of the proofs is that of *processor*. A *DP processor* is a function Proc that takes a DP problem and returns a (possibly empty) set of (possibly simpler) DP problems. Here 'simpler' usually means that fewer pairs are involved. A *sound* processor transforms DP problems in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. The processor is *complete* if there is an infinite minimal chain in the original DP problem if and only if there is an infinite minimal chain in the transformed problem. Soundness is essential for *proving* termination; completeness is required for *disproving* termination. The DP framework is formally introduced in the following theorem.

Theorem 6 (DP Framework [GTSF06, Thi07]) *Let $\mathcal{R}$ be a TRS. We construct a tree whose nodes are labeled as DP problems, "yes", or "no", and whose root is labeled with $(\mathsf{DP}(\mathcal{R}), \mathcal{R})$. For every inner node labeled with τ, there is a sound processor* Proc *that satisfies one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child, labeled with "no".*
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child, labeled with "yes".*
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$, *and the children of the node are labeled with the DP problems in* $\mathsf{Proc}(\tau)$.

If all leaves of the tree are labeled with "yes", then $\mathcal{R}$ is terminating. Otherwise, if there is a leaf labeled with "no" and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is nonterminating.

Processors are used in a *tree* to incrementally simplify the original DP problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. The trivial case of this *iterative* process comes when the set of pairs $\mathcal{P}$ becomes empty. In this way, we obtain a much more flexible framework to mechanize termination proofs and also to benefit from new future developments which could lead to the introduction of new processors.

1.4 Innermost Context-Sensitive Rewriting

Most computational systems whose operational principle is based on reducing expressions can be described and analyzed by using notions and techniques that come from the abstract model of TRSs [BN98, TeR03]. Such computational systems (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) often incorporate a predefined reduction strategy that is used to break down the nondeterminism that is inherent to reduction relations. Eventually, this can create problems, as each kind of strategy only behaves properly (i.e., it is normalizing, optimal, etc.) for particular classes of programs. One of the most commonly used strategies is the *innermost* one, in which only innermost redexes are reduced. Here, by an innermost redex, we mean a redex containing no other redex. The innermost strategy corresponds to call by value or *eager* computation, that is, the computational mechanism of several programming languages where the arguments of a function are always evaluated before the application of the function that uses them. It is well-known,

however, that programs written in eager programming languages frequently run into a nonterminating behavior if the programs have not carefully been written to avoid such problems. For this reason, the designers of these eager programming languages have also developed some features and language constructs aimed at giving the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to arguments of symbols) have been used in programming languages such as Clean [NSEP92], Haskell [HPW92], Lisp [McC60], Maude [CDEL+07], OBJ2 [FGJM85], OBJ3 [GWM+00], CafeOBJ [FN97], etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become 'more eager' and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become 'more lazy', thus (hopefully) avoiding nontermination. *Context-sensitive rewriting* (*CSR* [Luc98, Luc02]) is a restriction of rewriting that forbids reductions on some subexpressions and that has proved useful to model and analyze these programming language features at different levels, see, e.g., [BM06, DLM+04, DLM+08, GM04, Luc01b, LM09]. Such a restriction of the rewriting computations is formalized at a very simple syntactic level: that of the arguments of function symbols f in the signature Σ. A *replacement map* is a mapping $\mu : \Sigma \to \wp(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f in the signature Σ [Luc98]. We use them to discriminate the argument positions on which the rewriting steps are allowed. In *CSR*, we only rewrite μ-replacing subterms: every term t (as a whole) is μ-replacing by definition; and t_i (as well as all its μ-replacing subterms) is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$. The following example provides an illustrative case study involving the most well-known cases where *CSR* is useful for avoiding nontermination.

Example 7

The following nonterminating TRS $\mathcal{R}$ can be used to compute the list of prime numbers by using the well-known Erathostenes sieve[5] [GM99]:

$$\begin{aligned}
\texttt{primes} &\to \texttt{sieve}(\texttt{from}(\texttt{s}(\texttt{s}(0)))) \\
\texttt{from}(x) &\to x\texttt{:}\texttt{from}(\texttt{s}(x)) \\
\texttt{head}(x\texttt{:}y) &\to x \\
\texttt{tail}(x\texttt{:}y) &\to y \\
\texttt{if}(\texttt{true}, x, y) &\to x
\end{aligned}$$

[5] Without appropriate rules for defining symbol `div`, the TRS has no complete computational meaning. However, we take it here as given in [GM99] for the purpose of comparing different techniques for proving (innermost) termination of *CSR* by transformation.

$$\begin{aligned}
\texttt{if}(\texttt{false}, x, y) &\to y \\
\texttt{filter}(\texttt{s}(\texttt{s}(x)), y\texttt{:}z) &\to \texttt{if}(\texttt{div}(\texttt{s}(\texttt{s}(x)), y), \texttt{filter}(\texttt{s}(\texttt{s}(x)), z), y\texttt{:}\texttt{filter}(x, \texttt{sieve}(y)))) \\
\texttt{sieve}(x\texttt{:}y) &\to x\texttt{:}\texttt{filter}(x, \texttt{sieve}(y))
\end{aligned}$$

where $\mu(\texttt{:}) = \{1\}$ disallows reductions on the *list* part of the list constructor (`:`), thus making a kind of *lazy evaluation* of lists possible. Moreover, the definition of `if` encodes the expected behavior of conditional expressions: depending on the outcome (`true` or `false`) of the evaluation of the first argument b in a call $\texttt{if}(b, s, t)$, we would evaluate the second (s) or the third argument (t) of the call. However, in pure term rewriting, the three arguments b, s, and t in the call could be evaluated in any order, thus eventually leading to wasteful computations (for instance, one could evaluate s, t, and finally b!). In this case, we want `if` to behave in such a way that we only evaluate the second and third arguments after the evaluation of the first argument. We can achieve this behavior with *CSR* by using a replacement map μ such that $\mu(\texttt{if}) = \{1\}$, $\mu(\texttt{:}) = \{1\}$ and $\mu(f) = \{1, \ldots, \mathsf{ar}(f)\}$ for all $f \in \Sigma \setminus \{\texttt{if}, \texttt{:}\}$.

The replacement map in Example 7 exemplifies one of the most typical applications of context-sensitive rewriting as a computational mechanism. The declaration $\mu(\texttt{:}) = \{1\}$ disallows reductions on the *list* part of the list constructor (`:`), thus making a kind of *lazy evaluation* of lists possible. The other typical application is the declaration $\mu(\texttt{if}) = \{1\}$, which allows us to forbid reductions on the two *alternatives* s and t of *if-then-else* expressions $\texttt{if}(b, s, t)$ whereas it is still possible to perform reductions on the *Boolean* part b, as required to implement the usual semantics of the operator.

Our focus is on termination of *innermost context-sensitive rewriting* (i.e., the variant of *CSR* where only the deepest μ-replacing redexes are contracted). Termination of innermost context-sensitive rewriting has been proved useful for proving termination of programs in programming languages like Maude and OBJ*, which permit the program execution to be controlled by means of such context-sensitive annotations [Luc01a, Luc01b]. Techniques for proving termination of innermost *CSR* were first investigated in [GM02a, GM02b, Luc01a]. These papers, however, only consider *transformational* techniques, where the original CS-TRS $(\mathcal{R}, \mu)$ is transformed into a TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation which has been used) whose *innermost* termination implies the innermost termination of *CSR* for $(\mathcal{R}, \mu)$.

Example 8

Consider the following system obtained after applying the only correct and complete existing transformation [GM02a] for proving innermost μ-termination

to the CS-TRS in Example 7:

$$
\begin{aligned}
\mathtt{active}(\mathtt{primes}) &\to \mathtt{mark}(\mathtt{sieve}(\mathtt{from}(\mathtt{s}(\mathtt{s}(0)))))\\
\mathtt{active}(\mathtt{from}(x)) &\to \mathtt{mark}(x\mathtt{:}\mathtt{from}(\mathtt{s}(x)))\\
\mathtt{active}(\mathtt{head}(x\mathtt{:}y)) &\to \mathtt{mark}(x)\\
\mathtt{active}(\mathtt{tail}(x\mathtt{:}y)) &\to \mathtt{mark}(y)\\
\mathtt{active}(\mathtt{if}(\mathtt{true},x,y)) &\to \mathtt{mark}(x)\\
\mathtt{active}(\mathtt{if}(\mathtt{false},x,y)) &\to \mathtt{mark}(y)\\
\mathtt{active}(\mathtt{filter}(\mathtt{s}(\mathtt{s}(x)),y\mathtt{:}z)) &\to \mathtt{mark}(\mathtt{if}(\mathtt{div}(\mathtt{s}(\mathtt{s}(x)),y),\mathtt{filter}(\mathtt{s}(\mathtt{s}(x)),z),\\
&\qquad\qquad y\mathtt{:}\mathtt{filter}(x,\mathtt{sieve}(y))))\\
\mathtt{active}(\mathtt{sieve}(x\mathtt{:}y)) &\to \mathtt{mark}(x\mathtt{:}\mathtt{filter}(x,\mathtt{sieve}(y)))\\
\mathtt{mark}(\mathtt{primes}) &\to \mathtt{active}(\mathtt{primes})\\
\mathtt{mark}(\mathtt{sieve}(x)) &\to \mathtt{active}(\mathtt{sieve}(\mathtt{mark}(x)))\\
\mathtt{mark}(\mathtt{from}(x)) &\to \mathtt{active}(\mathtt{from}(\mathtt{mark}(x)))\\
\mathtt{mark}(\mathtt{s}(x)) &\to \mathtt{active}(\mathtt{s}(\mathtt{mark}(x)))\\
\mathtt{mark}(0) &\to \mathtt{active}(0)\\
\mathtt{mark}(x1\mathtt{:}x2) &\to \mathtt{active}(\mathtt{mark}(x1)\mathtt{:}x2)\\
\mathtt{mark}(\mathtt{head}(x)) &\to \mathtt{active}(\mathtt{head}(\mathtt{mark}(x)))\\
\mathtt{mark}(\mathtt{tail}(x)) &\to \mathtt{active}(\mathtt{tail}(\mathtt{mark}(x)))\\
\mathtt{mark}(\mathtt{if}(x1,x2,x3)) &\to \mathtt{active}(\mathtt{if}(\mathtt{mark}(x1),x2,x3))\\
\mathtt{mark}(\mathtt{true}) &\to \mathtt{active}(\mathtt{true})\\
\mathtt{mark}(\mathtt{false}) &\to \mathtt{active}(\mathtt{false})\\
\mathtt{mark}(\mathtt{filter}(x1,x2)) &\to \mathtt{active}(\mathtt{filter}(\mathtt{mark}(x1),\mathtt{mark}(x2)))\\
\mathtt{mark}(\mathtt{div}(x1,x2)) &\to \mathtt{active}(\mathtt{div}(\mathtt{mark}(x1),\mathtt{mark}(x2)))\\
\mathtt{sieve}(\mathtt{mark}(x)) &\to \mathtt{sieve}(x)\\
\mathtt{sieve}(\mathtt{active}(x)) &\to \mathtt{sieve}(x)\\
\mathtt{from}(\mathtt{mark}(x)) &\to \mathtt{from}(x)\\
\mathtt{from}(\mathtt{active}(x)) &\to \mathtt{from}(x)\\
\mathtt{s}(\mathtt{mark}(x)) &\to \mathtt{s}(x)\\
\mathtt{s}(\mathtt{active}(x)) &\to \mathtt{s}(x)\\
\mathtt{mark}(x1)\mathtt{:}x2 &\to x1\mathtt{:}x2\\
x1\mathtt{:}\mathtt{mark}(x2) &\to x1\mathtt{:}x2\\
\mathtt{active}(x1)\mathtt{:}x2 &\to x1\mathtt{:}x2\\
x1\mathtt{:}\mathtt{active}(x2) &\to x1\mathtt{:}x2\\
\mathtt{head}(\mathtt{mark}(x)) &\to \mathtt{head}(x)\\
\mathtt{head}(\mathtt{active}(x)) &\to \mathtt{head}(x)
\end{aligned}
$$

$$
\begin{array}{rcl}
\mathtt{tail}(\mathtt{mark}(x)) & \to & \mathtt{tail}(x) \\
\mathtt{tail}(\mathtt{active}(x)) & \to & \mathtt{tail}(x) \\
\mathtt{if}(\mathtt{mark}(x1), x2, x3) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{if}(x1, \mathtt{mark}(x2), x3) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{if}(x1, x2, \mathtt{mark}(x3)) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{if}(\mathtt{active}(x1), x2, x3) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{if}(x1, \mathtt{active}(x2), x3) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{if}(x1, x2, \mathtt{active}(x3)) & \to & \mathtt{if}(x1, x2, x3) \\
\mathtt{filter}(\mathtt{mark}(x1), x2) & \to & \mathtt{filter}(x1, x2) \\
\mathtt{filter}(x1, \mathtt{mark}(x2)) & \to & \mathtt{filter}(x1, x2) \\
\mathtt{filter}(\mathtt{active}(x1), x2) & \to & \mathtt{filter}(x1, x2) \\
\mathtt{filter}(x1, \mathtt{active}(x2)) & \to & \mathtt{filter}(x1, x2) \\
\mathtt{div}(\mathtt{mark}(x1), x2) & \to & \mathtt{div}(x1, x2) \\
\mathtt{div}(x1, \mathtt{mark}(x2)) & \to & \mathtt{div}(x1, x2) \\
\mathtt{div}(\mathtt{active}(x1), x2) & \to & \mathtt{div}(x1, x2) \\
\mathtt{div}(x1, \mathtt{active}(x2)) & \to & \mathtt{div}(x1, x2)
\end{array}
$$

AProVE, which is the most powerful tool for proving innermost termination of standard rewriting[6], fails when trying to prove innermost termination of this system. Therefore, innermost μ-termination of Example 7 cannot be proven by using the transformational approach of [GM02a] since the other two correct existing transformations for proving innermost termination of *CSR* [GM02a] also fail.

We started the adaptation of the DP framework to termination of *CSR* in [AGL06]. Proving innermost termination of rewriting is often easier than proving termination of rewriting [AG00] and, for some relevant classes of TRSs, innermost termination of rewriting is even equivalent to termination of rewriting [Gra95, Gra96]. In [GM02a, GL02a] it is proved that the equivalence between termination of innermost *CSR* and termination of *CSR* holds in some interesting cases (e.g., for *orthogonal* CS-TRSs). From the termination point of view, Example 7 is interesting because, since its introduction in Giesl and Middeldorp's paper [GM99], no automatic proof of (innermost) μ-termination has been reported by using transformations. In sharp contrast, innermost termination of *CSR* for this TRS and replacement map μ is easily proved by using

[6] See [MZ07, Wal09] for a summary of the results of the Termination Competition in 2007 and 2008.

the techniques developed in this thesis. In particular, the innermost context-sensitive dependency graph contains no cycle, thus, innermost μ-termination can be proved trivially. Moreover, since the system is orthogonal, we can also conclude that the system is μ-terminating. During the last few years, we have continued the research about how to prove termination of context-sensitive rewriting by using dependency pairs, since they have proven to be one of the most powerful techniques for proving termination of unrestricted rewriting. In [AGL06, AGL10], we define the notion of context-sensitive dependency pairs following the approach of [HM04], which consists of considering the structure of the infinite rewrite sequences starting from minimal nonterminating terms. This allows us to extend the DP framework to deal with proofs of termination of *CSR* and define our context-sensitive dependency pair framework (CSDP framework [AGL10]), which has also been recently improved in [GL10]. Therefore, all the advantages and improvements on this research can also be taken into account in innermost context-sensitive rewriting, improving our previous results in this field [AL07].

1.5 Rewriting Modulo Equational Theories

Rewriting with rules R modulo axioms E is a widely used technique in both rule-based programming languages and in automated deduction. Consequently, termination of rewriting modulo specific equational axioms E (e.g., associativity-commutativity, AC) has been studied. Methods for proving termination of rewriting systems modulo AC-axioms are known and even implemented. Several works have tried to adapt the DP approach [AG00] to rewriting modulo *associative and commutative* theories [KNT06, KT01, Kus00, MU98, MU04]. The corresponding proof methods, however, cannot be applied to commonly occurring combinations of axioms that fall outside their scope.

Example 9

Consider the (order-sorted) TRS specified in Maude in Figure 1.2. It has four sorts: `Bool`, `Nat`, `List`, and `Set`, with `Nat` included in both `List` and `Set` as a subsort. That is, a natural number n is simultaneously regarded as a list of length 1 and as a singleton set. The terms of each sort are, respectively, Booleans, natural numbers (in Peano notation), lists of natural numbers, and finite sets of natural numbers. The equations in this module then define various functions such as `_and_` and `_or_`, a function `list2set` associating to each list its corresponding set, the set membership predicate `_in_`, and an equality predicate `_==_` on lists. Furthermore, the idempotency of set union is specified by the first equation. All these equations rewrite terms *modulo* the equational

```
fmod LIST&SET is
  sorts Bool Nat List Set .
  subsorts Nat < List Set .
  ops true false : -> Bool .
  ops _and_ _or_ : Bool Bool -> Bool [assoc comm] .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
  op _;_ : List List -> List [assoc] .
  op null : -> Set .
  op __ : Set Set -> Set [assoc comm] .
  op _in_ : Nat Set -> Bool .
  op _==_ : List List -> Bool [comm] .
  op list2set : List -> Set .
  var  B : Bool .              vars N M : Nat .
  vars L L' : List .           var  S : Set .
  eq N N = N .
  eq true and B = B .          eq false and B = false .
  eq true or B = true .        eq false or B = B .
  eq 0 == s N = false .        eq s N == s M = N == M .
  eq N ; L == M = false .      eq N ; L == M ; L' = (N == M) and L == L' .
  eq L == L = true .
  eq list2set(N) = N .         eq list2set(N ; L) = N list2set(L) .
  eq N in null = false .       eq N in M S = (N == M) or N in S .
endfm
```

Figure 1.2: Example in Maude syntax [DLM09b]

axioms declared in the module. Specifically, `_and_` and `_or_` have been declared associative and commutative with the `assoc` and `comm` keywords, the list concatenation operator `_;_` has been declared associative using the `assoc` keyword; the set union operator `__` has been declared associative and commutative using the `assoc` and `comm` keywords; and the `_==_` equality predicate has been declared commutative using the `comm` keyword. The succinctness of this specification is precisely due to the power of rewriting modulo axioms, which typically uses considerably fewer rules than standard rewriting.

Methods for proving termination of AC-theories could not be applied to prove termination of the TRS in Figure 1.2 (we would not care about sort information here), where we have an *arbitrary combination* of associative and/or commutative axioms which we call an $A \vee C$*-rewrite theory* [ALM10].

Giesl and Kapur generalized the previous works on AC-termination with dependency pairs to deal with more general kinds of equational theories E satisfying some restrictions [GK01]. In principle, the $A \vee C$-theories that we

investigate here fit the main outlines of Giesl and Kapur's approach. However, [GK01] did not provide any definition of *minimal chain*, which is needed for further developments in the DP framework. In this thesis, we address this problem and provide an appropriate notion of minimal nonterminating term and of chain for $A \vee C$-theories which is used to develop an $A \vee C$-dependency pair framework for proving termination of $A \vee C$-rewrite theories.

1.6 Plan of the Thesis

This PhD thesis has been developed under the "publications format" that has recently been adopted for the presentation of PhD thesis in Spanish universities[7]. According to this format, a PhD thesis can consist of a collection of papers that have previously been published in relevant venues, together with an extended summary of the research performed. In the first part of this PhD thesis, we present the summary itself which is divided into two parts that independently treat two important problems related to proving termination of Maude programs. Part I contains the development of techniques for proving termination of innermost context-sensitive rewriting, and Part II deals with proving termination of rewriting modulo associative and/or commutative axioms. Both extend the DP framework to automate proofs of termination of each domain. After the summary of the research, the list of publications that develop the material in this thesis is presented, accompanied by the full text for each of them as originally published.

In Chapter 2, we present some preliminary definitions and results, which are used in the main text of the thesis and also in the attached papers.

- The material in Part I, *Termination of Innermost Context-Sensitive Rewriting*, is structured as follows:
  - In Chapter 3, we investigate the structure of infinite innermost context-sensitive rewrite sequences. This analysis is essential in order to provide an appropriate definition of innermost context-sensitive dependency pair, and the related notions of innermost μ-chains, graph, etc. We provide appropriate notions of *minimal* innermost non-μ-terminating terms and introduce the main properties of these terms. As in *CSR*, this analysis leads to the notion

[7] Article 21 from "Real Decreto 1393/2007" and Item 4 from "Normativa por la que se establece el procedimiento regulador para la elaboración y defensa de las tesis doctorales en la Universidad Politécnica de Valencia (Aprobada en Comisión de Doctorado de fecha 23 de octubre de 2008)"

of *hidden term*, which is essential for the appropriate treatment of our dependency pairs.

- In Chapter 4, we define the notions of *innermost context-sensitive dependency pair* (ICSDP) and *innermost context-sensitive chain of pairs* and show how to use them to *characterize* innermost termination of *CSR*.
- Chapter 5 introduces the general framework to compute and use innermost context-sensitive dependency pairs for proving innermost termination of *CSR*. It provides an adaptation of the DP framework to innermost *CSR* by defining appropriate notions of *CS problem* and *CS processor*, which rely on the notions and results previously investigated. We also relate innermost termination of *CSR* with μ-termination as well as with full rewriting.
- Chapter 6 introduces several processors to be used in our ICSDP framework. It includes the innermost context-sensitive (dependency) graph, use of reduction pairs (with and without usable rules and usable arguments), narrowing, etc.

Part I ends with an experimental evaluation of our techniques in Chapter 7. Chapter 8 relates this work to previous work and summarizes the main contributions.

- The material in Part II, *Termination of $A\vee C$-Rewriting*, is structured as follows:
  - Chapter 9 investigates the drawbacks of previous notions of minimal E-nonterminating term when modeling infinite $A\vee C$-rewrite sequences. Then, we introduce the notion of *stably minimal* E-nonterminating term, which is the basis of our development, and we investigate the structure of infinite $A\vee C$-sequences starting from such stably minimal terms.
  - Chapter 10 uses these results to formalize our notions of $A\vee C$-dependency pairs and of minimal chains and shows how to use them to *characterize* termination of $A\vee C$-rewrite theories.
  - Chapter 11 extends the DP framework to $A\vee C$-termination by defining appropriate notions of *$A\vee C$ problem* and *$A\vee C$ processor* that rely on the results previously obtained.
  - Chapter 12 develops several specific processors to be used in our $A\vee C$ framework, including the use of usable rules and equations together with reduction orders.

Part II ends with an experimental evaluation of our techniques in Chapter 13. Chapter 14 relates this work to previous work and summarizes the main contributions.

Finally, Chapter 15 briefly explains some details about our termination tool MU-TERM and the Maude termination tool MTT that proves termination of Maude programs by transformation. Chapter 16 concludes the thesis.

The publications associated with this thesis are the following (in chronological order):

1. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 2006.

2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph**. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007. Selected papers from the 6th Spanish Conference on Programming and Languages, PROLE'06.

3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with** MU-TERM. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007. Selected papers from the 6th Spanish Conference on Programming and Languages, PROLE'06.

4. B. Alarcón and S. Lucas. **Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs**. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 73–87, Springer-Verlag, 2007.

5. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider -Kamp, and R. Thiemann. **Improving Context-Sensitive Dependency Pairs**. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of XV International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *Lecture Notes in Computer Science*, pages 636–651. Springer-Verlag, 2008.

6. B. Alarcón, and S. Lucas. **Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting**. *Electronic Notes in Theoretical Computer Science*, 248:3-17, 2009. Selected papers from the 8th Spanish Conference on Programming and Languages, PROLE'08.

7. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. *Information and Computation*, 208:922–968, 2010.

8. B. Alarcón, S. Lucas and J. Meseguer. **A Dependency Pair Framework for $A \vee C$-Termination**. In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 35–51. Springer-Verlag, 2010.

9. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. **Proving Termination Properties with** MU-TERM. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology and Software Technology, AMAST'10*, volume 6486 of *Lecture Notes in Computer Science*, pages 201–208. Springer-Verlag, 2010.

10. B. Alarcón, and S. Lucas. **Innermost Termination of Context-Sensitive Rewriting**. Technical Report, http://hdl.handle.net/10251/10796, DSIC-ELP, UPV, 2011.

11. B. Alarcón, R. Gutiérrez, S. Lucas, and J. Meseguer. **A Dependency Pair Framework for $A \vee C$-Termination**. Technical Report, http://hdl.handle.net/10251/10797, DSIC-ELP, UPV, 2011.

2

Preliminaries

This chapter presents a number of definitions and notations about term rewriting that are used in this thesis and also in the reference papers that are attached at the end. More details and missing notions can be found in [BN98, Ohl02, TeR03].

2.1 Abstract Reduction Systems

Let A be a set and $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$ be a binary relation on A. An *abstract reduction system* is a pair (A, R). If $a, b \in \mathsf{A}$, we write $a \mathsf{R} b$ and say that a *reduces to* b *in one step*, instead of $(a, b) \in \mathsf{R}$. An R-*reduction sequence* is a finite or infinite sequence $a_0 \mathsf{R} a_1 \mathsf{R} a_2 \mathsf{R} a_3 \mathsf{R} \cdots$. We denote the transitive closure of R by R^+, its reflexive closure by $\mathsf{R}^=$, and its reflexive and transitive closure by R^*. An element $a \in \mathsf{A}$ is called an R-*normal form* if there exists no b such that $a \mathsf{R} b$. We say that R is *terminating* (also known as *strongly normalizing*, *well-founded* , or *noetherian*) if there is no infinite reduction sequence $a_1 \mathsf{R} a_2 \mathsf{R} a_3 \cdots$. A reflexive and transitive relation R is a *quasi-order* .

2.2 Signatures, Terms, and Positions

A *signature* Σ is a countable set of function symbols $\{\mathtt{f}, \mathtt{g}, \mathtt{if}, \mathtt{from}, \mathtt{true} \ldots\}$, each having a fixed number of arguments called *arity* and given by a mapping $\mathsf{ar} : \Sigma \to \mathbb{N}$. We often write Σ^k to refer to the symbols of Σ whose arity is k. Function symbols with arity 0 are called constants. And $\mathcal{X}$ denotes a countable set of variables. The set of terms $\mathcal{T}(\Sigma, \mathcal{X})$, built from Σ and $\mathcal{X}$, is inductively defined as follows:

- $x \in \mathcal{T}(\Sigma, \mathcal{X})$ if $x \in \mathcal{X}$, and
- $f(t_1, \ldots, t_k) \in \mathcal{T}(\Sigma, \mathcal{X})$ if $t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, $f \in \Sigma$ and $\mathsf{ar}(f) = k$.

The set of variables of a term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is denoted as $\mathcal{V}ar(t)$. A term t is *ground* if it contains no variable. A term is said to be *linear* if it has no multiple occurrences of the same variable.

Labelled trees provide a natural way of representing terms. Leaves are labelled with variables from $\mathcal{X}$ or constant symbols from Σ^0. Inner nodes are labelled with function symbols $f \in \Sigma \setminus \Sigma^0$ and with $\mathsf{ar}(f)$ subtrees. *Positions* $p, q, \ldots$ are chains of positive natural numbers that are used to address subterms of t. The root position, referring to the whole term, corresponds to an empty chain and is denoted by Λ. Given positions p, q, their concatenation is denoted as $p.q$. Positions are ordered by the standard prefix order: $p \leq q$ if $\exists q'$ such that $q = p.q'$. If p is a position, and Q is a set of positions, then $p.Q = \{p.q \mid q \in Q\}$ is the set of positions obtained from Q by adding a prefix p to each position $q \in Q$. The set of positions of a term t is $\mathcal{P}os(t)$. Given a signature Σ and a set of variables $\mathcal{X}$, the set of positions of nonvariable symbols occurring in t is denoted as $\mathcal{P}os_\Sigma(t)$, and $\mathcal{P}os_\mathcal{X}(t)$ is the set of positions of variable occurrences in t. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s.

A *context* is a term $C[\,] \in \mathcal{T}(\Sigma \cup \{\Box\}, \mathcal{X})$ with zero or more 'holes'. A hole $\Box$ is a fresh constant symbol. We write $C[\,]_p$ to denote that there is a hole $\Box$ at position p of $C[\,]$. Generally, $C[\,]$ is written to denote an arbitrary context; we make the position of the hole explicit only if necessary. $C[\,] = \Box$ is called the *empty* context.

We write $s \trianglerighteq t$ to denote that t *is a subterm of* s, i.e. $t = s|_p$ for some $p \in \mathcal{P}os(s)$; we write $s \triangleright t$ if $s \trianglerighteq t$ and $s \neq t$ (i.e., t is a *strict* subterm of s). We write $s \not\trianglerighteq t$ and $s \not\triangleright t$ to negate the corresponding properties. The symbol labeling the root of t is denoted as $\mathsf{root}(t)$.

2.3 Substitutions, Renamings, and Unifiers

A substitution is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$ where the set $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ is called the *domain* of σ. In this thesis, we do *not* impose that the domain of the substitutions be finite. A substitution can be extended to a function from terms to terms by $\sigma(f(t_1, \ldots, t_k)) = f(\sigma(t_1), \ldots, \sigma(t_k))$ for each k-ary function symbol $f \in \Sigma$ and terms $t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$.

We say that term t *matches* s, if s is an instance of t, i.e., there is a substitution σ such that $\sigma(t) = s$. A *renaming* is an injective substitution ρ such that $\rho(x) \in \mathcal{X}$ for all $x \in \mathcal{X}$.

A substitution σ such that $\sigma(s) = \sigma(t)$ for two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$ is called a *unifier* of s and t; it is also said that s and t unify (with substitution

σ). If two terms s and t unify, then there is a unique (up to renaming of variables) *most general unifier (mgu)* θ such that, for every other unifier τ, there is a substitution σ such that $\theta \circ \tau = \sigma$.

2.4 Binary Relations over Terms

A relation $\mathsf{R} \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ on terms is *stable under substitutions* if for all terms $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$, and substitutions σ, we have $\sigma(s)\,\mathsf{R}\,\sigma(t)$ whenever $s\,\mathsf{R}\,t$.

A relation $\mathsf{R} \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ on terms is *monotonic* (also called stable under contexts) if for all terms $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$ and context $C[\,]$, we have $C[s]\,\mathsf{R}\,C[t]$ whenever $s\,\mathsf{R}\,t$.

Monotonicity can also be expressed in the following way: a relation $\mathsf{R} \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ on terms is *monotonic* if for all symbols $f \in \Sigma$, arguments i, $1 \leq i \leq k$, and terms $s, t, t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, we have

$$f(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_k)\,\mathsf{R}\,f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_k)$$

whenever $s\,\mathsf{R}\,t$.

2.5 Rewrite Systems and Term Rewriting

A *rewrite rule* is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\Sigma, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l, and the right-hand side (*rhs*) of the rule is r. A rewrite rule $l \to r$ is said to be *collapsing* if $r \in \mathcal{X}$. A *term rewriting system* (TRS) is a pair $\mathcal{R} = (\Sigma, R)$, where R is a set of rewrite rules. Given TRSs $\mathcal{R} = (\Sigma, R)$ and $\mathcal{R}' = (\Sigma', R')$, we let $\mathcal{R} \cup \mathcal{R}'$ be the TRS $(\Sigma \cup \Sigma', R \cup R')$. An instance $\sigma(l)$ of a *lhs* l of a rule is called a *redex*. Given $\mathcal{R} = (\Sigma, R)$, we consider Σ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{D}$ of symbols $\mathsf{c} \in \mathcal{C}$ (called *constructors*) and symbols $f \in \mathcal{D}$ (called *defined functions*), where $\mathcal{D} = \{\mathsf{root}(l) \mid l \to r \in R\}$ and $\mathcal{C} = \Sigma \setminus \mathcal{D}$. For simplicity, we often write $l \to r \in \mathcal{R}$ instead of $l \to r \in R$ to express that the rule $l \to r$ is a rule of $\mathcal{R}$.

A term $s \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to t (at position p), written $s \xrightarrow{p}_{\mathcal{R}} t$ (or just $s \to_{\mathcal{R}} t$, or $s \to t$), if $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(s)$, and substitution σ. We write $s \xrightarrow{>p}_{\mathcal{R}} t$ if $s \xrightarrow{q}_{\mathcal{R}} t$ for some $q > p$. A TRS $\mathcal{R}$ is terminating if its one step rewrite relation $\to_{\mathcal{R}}$ is terminating.

2.6 Innermost Rewriting

A term is a normal form if it contains no redex. A substitution σ is normalized if $\sigma(x)$ is a normal form for all $x \in \mathcal{D}om(\sigma)$. A term $f(t_1, \ldots, t_k)$ is argument normalized if t_i is a normal form for all $1 \leq i \leq n$. An innermost redex is an argument normalized redex. A term s rewrites innermost to t, written $s \to_{\mathsf{i}} t$, if $s \to t$ at position p and $s|_p$ is an innermost redex. Let $\mathcal{R}$ be a TRS. For any symbol f let $Rls(\mathcal{R}, f)$ be the set of rules $l \to r$ defining f and such that the left-hand sides l are argument normalized. For any term t the set of *usable rules* $\mathbf{U}(\mathcal{R}, t)$ is as follows:

$$\begin{array}{lcl} \mathbf{U}(\mathcal{R}, x) & = & \varnothing \\ \mathbf{U}(\mathcal{R}, f(t_1, \ldots, t_n)) & = & Rls(\mathcal{R}, f) \cup \bigcup_{i \in \mathsf{ar}(f)} \mathbf{U}(\mathcal{R}', t_i) \cup \bigcup_{l \to r \in Rls(\mathcal{R}, f)} \mathbf{U}(\mathcal{R}', r) \end{array}$$

where $\mathcal{R}' = \mathcal{R} - Rls(\mathcal{R}, f)$.

2.7 (Innermost) Context-Sensitive Rewriting

A mapping $\mu : \Sigma \to \wp(\mathbb{N})$ is a *replacement map* (or Σ-map) if for all symbols $f \in \Sigma$, $\mu(f) \subseteq \{1, \ldots, \mathsf{ar}(f)\}$ [Luc98]. Let M_Σ be the set of all Σ-maps (or $M_\mathcal{R}$ for the Σ-maps of a TRS $\mathcal{R} = (\Sigma, R)$). Let $\mu_\top$ be the replacement map given by $\mu_\top(f) = \{1, \ldots, \mathsf{ar}(f)\}$ for all $f \in \Sigma$ (i.e., no replacement restrictions are specified).

A binary relation R on terms is *μ-monotonic* if for all symbols $f \in \Sigma$, arguments $i \in \mu(f)$, and terms $s, t, t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, we have

$$f(t_1, \ldots, t_{i-1}, s, t_{i+1} \ldots, t_k) \; \mathsf{R} \; f(t_1, \ldots, t_{i-1}, t, t_{i+1} \ldots, t_k)$$

whenever $s \; \mathsf{R} \; t$.

The set of *μ-replacing positions* $\mathcal{P}os^\mu(t)$ of $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(\mathsf{root}(t))} i.\mathcal{P}os^\mu(t|_i)$, if $t \notin \mathcal{X}$. When no replacement map is made explicit, the μ-replacing positions are often called *active*; and the non-μ-replacing ones are often called *frozen*. The μ-replacing subterm relation $\trianglerighteq_\mu$ is given by $s \trianglerighteq_\mu t$ if there is $p \in \mathcal{P}os^\mu(s)$ such that $t = s|_p$. We write $s \triangleright_\mu t$ and say that t is a *strict* μ-replacing subterm of s if $s \trianglerighteq_\mu t$ and $s \neq t$. We write $s \triangleright_{\not\mu} t$ if t is a non-μ-replacing (hence, strict) subterm of s, i.e., there is $p \in \mathcal{P}os(s) \setminus \mathcal{P}os^\mu(s)$ such that $t = s|_p$. The set of *μ-replacing* variables of a term t, i.e., variables occurring at some μ-replacing position in t, is $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid t \trianglerighteq_\mu x\}$. The set of non-$\mu$-replacing variables of t, i.e., variables occurring at some frozen position

in t, is $\mathcal{V}ar^{\not\mu}(t) = \{x \in \mathcal{V}ar(t) \mid t \rhd_{\not\mu} x\}$. Note that $\mathcal{V}ar^{\mu}(t)$ and $\mathcal{V}ar^{\not\mu}(t)$ do not need to be disjoint.

A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a CS-TRS . In *context-sensitive rewriting*, (only) *μ-replacing* redexes are contracted: s μ-rewrites to t, written $s \overset{p}{\hookrightarrow}_{\mathcal{R},\mu} t$ (or $s \hookrightarrow_{\mathcal{R},\mu} t$, $s \hookrightarrow_{\mu} t$, and even $s \hookrightarrow t$), if $s \overset{p}{\rightarrow}_{\mathcal{R}} t$ and $p \in \mathcal{P}os^{\mu}(t)$.

Example 10

Consider $\mathcal{R}$ and μ as in Example 7. Then, we have:

$$\underline{\texttt{from(0)}} \hookrightarrow_{\mu} \texttt{0:}\underline{\texttt{from(s(0))}} \not\hookrightarrow_{\mu} \texttt{0:(s(0):from(s(s(0))))}$$

since the second argument of (:) is not μ-replacing, $2 \notin \mathcal{P}os^{\mu}(\texttt{0:from(s(0))})$, and the redex $\texttt{from(s(0))}$ cannot be μ-rewritten.

A term t is *μ-terminating* (or $(\mathcal{R}, \mu)$-terminating), if we want to explicitly refer to the involved TRS $\mathcal{R}$) if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \cdots \hookrightarrow_{\mathcal{R},\mu} t_n \hookrightarrow_{\mathcal{R},\mu} \cdots$ starting from t. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mu}$ is terminating.

A *μ-normal form* is a term which cannot be μ-rewritten. Let $\mathsf{NF}_{\mu}(\mathcal{R})$ (or just NF_{μ} if no confusion arises) be the set of μ-normal forms of a CS-TRS $(\mathcal{R}, \mu)$. A substitution σ is *μ-normalized* if $\sigma(x)$ is a μ-normal form for all $x \in \mathcal{D}om(\sigma)$. A term $t = f(t_1, \ldots, t_k)$ is *argument μ-normalized* if t_i is a μ-normal form for all $i \in \mu(f)$. A *μ-innermost redex* is an argument μ-normalized redex, i.e., $t = \sigma(l)$ for some substitution σ and rule $l \to r \in \mathcal{R}$ and for all $p \in \mathcal{P}os^{\mu}(t - \Lambda)$, $t|_p \in \mathsf{NF}_{\mu}$. A term s innermost μ-rewrites to t, written $s \hookrightarrow_{\mathsf{i}} t$, if $s \overset{p}{\rightarrow}_{\mathcal{R}} t$, $p \in \mathcal{P}os^{\mu}(s)$, and $s|_p$ is a μ-innermost redex. Let innermost μ-rewriting below the root be $\overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} = (\overset{>\Lambda}{\hookrightarrow} \cap \hookrightarrow_{\mathsf{i}})$. Termination of *CSR* is fully captured by the so-called *μ-reduction orders*, i.e., well-founded, stable orders $\sqsupset$ which are μ-monotonic. A TRS $\mathcal{R}$ is *innermost μ-terminating* if $\hookrightarrow_{\mu,\mathsf{i}}$ is terminating. We write $s \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} t$ if $s \hookrightarrow^{*}_{\mathcal{R},\mu,\mathsf{i}} t$ and $t \in \mathsf{NF}_{\mu}$.

2.8 Narrowing

Narrowing combines rewriting with unification. Given a TRS $\mathcal{R} = (\Sigma, R)$, a term s *narrows* to a term t (written $s \overset{p}{\rightsquigarrow}_{\mathcal{R},\theta} t$, $s \rightsquigarrow_{\mathcal{R},\theta} t$, or $s \rightsquigarrow_{\theta} t$), if there is a nonvariable position $p \in \mathcal{P}os_{\Sigma}(s)$ and a rule $l \to r \in R$ (sharing no variable with s) such that $s|_p$ and l unify with the most general unifier θ and $t = \theta(s[r]_p)$. A term s *μ-narrows* to a term t (written $s \overset{p}{\rightsquigarrow}_{\mathcal{R},\mu,\theta} t$, $s \rightsquigarrow_{\mathcal{R},\mu,\theta} t$, or $s \rightsquigarrow_{\mu,\theta} t$), if $s \overset{p}{\rightsquigarrow}_{\mathcal{R},\theta} t$ and $p \in \mathcal{P}os^{\mu}_{\Sigma}(s)$.

2.9 Rewriting Modulo Equational Theories

Given a set of equations E, we write $s \overset{p}{\vdash\!\dashv}_E t$ (a single 'equational step') if there is a position $p \in \mathcal{P}os(s)$, an equation $u = v \in E$ and a substitution σ such that $s|_p = \sigma(u)$ and $t|_p = \sigma(v)$, or $s|_p = \sigma(v)$ and $t|_p = \sigma(u)$ (we write $s \vdash\!\dashv_E t$ if position p is not relevant). Note that $\vdash\!\dashv_E$ is a symmetric relation. Then, $\sim_E$ is the reflexive and transitive closure of $\vdash\!\dashv_E$; we have the following equivalence that will be useful in our development:

$$\sim_E = \vdash\!\dashv_E^* = (\overset{\Lambda}{\vdash\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\dashv}_E)^*.$$

We also write $s \overset{>\Lambda}{\sim}_E t$ iff $s = f(s_1, \ldots, s_k)$, $t = f(t_1, \ldots, t_k)$ and $s_i \sim_E t_i$ for all i, $1 \leq i \leq k$. We define $s \overset{\Lambda}{\sim}_E t$ as the reflexive and transitive closure of $\overset{\Lambda}{\vdash\!\dashv}_E$.

Given a rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where R is a set of rewrite rules and E is a set of equational axioms, we write $s \to_{R/E} t$ if there exist u, v such that $s \sim_E u$, $u \to_R v$, and $v \sim_E t$.We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is *E-terminating* , iff $\to_{R/E}$ is terminating. In general, given terms s and t, the problem of checking whether $s \to_{R/E} t$ holds is undecidable: in order to check whether $s \to_{R/E} t$ we have to search through the possibly infinite equivalence classes $[s]_E$ and $[t]_E$ to see whether a matching is found for a subterm of some $u \in [s]_E$ and the result of rewriting u belongs to the equivalence class $[t]_E$. For this reason, a much simpler relation $\to_{R,E}$ is defined, which becomes decidable if an E-matching algorithm exists. For any terms s, t, $s \to_{R,E} t$ holds iff there is a position p in s, a rule $l \to r$ in R, and a substitution σ such that $s|_p \sim_E \sigma(l)$ and $t = s[\sigma(r)]_p$ (see [PS81]). This relation only allows applying rules from R in redexes at positions equal or above of positions of terms where equations from E have been applied. We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is *(R, E)-terminating*, if $\to_{R,E}$ is terminating. In the following, we assume that E and R are finite sets of equations and rules, respectively. This requirement is needed, for instance, to ensure that the interpretation used when dealing with usable rules and equations is well-defined (see [AGLM11]).

Part I

Termination of Innermost Context-Sensitive Rewriting

3

Infinite Innermost Context-Sensitive Rewrite Sequences

In the following, we show how to adapt our results about the structure of infinite context-sensitive rewrite sequences [AGL10, Section 3] to the innermost setting. As for context-sensitive rewriting, we analyze the structure of infinite sequences starting from minimal terms. Most results are only slightly different from the ones obtained for context-sensitive rewriting, and, therefore, we comment on the differences only (for full details see [AL11] and [AGL10]). Major differences come from particularities of reductions under an innermost strategy. In the innermost (context-sentive) setting, matching substitutions are always (μ)-normalized. In some cases, they bring us some advantages over the case of 'free' reductions. In standard rewriting, given a TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$, the *minimal* nonterminating terms associated to $\mathcal{R}$ are nonterminating terms t whose proper subterms u (i.e., $t \rhd u$) are terminating; $\mathcal{T}_\infty$ is the set of minimal nonterminating terms associated to $\mathcal{R}$ [HM04, HM07]. Minimal nonterminating terms have two important properties:

1. Every nonterminating term s contains a minimal nonterminating term $t \in \mathcal{T}_\infty$ (i.e., $s \trianglerighteq t$), and
2. minimal nonterminating terms t are always rooted by a *defined* symbol $f \in \mathcal{D}$: $\forall t \in \mathcal{T}_\infty, root(t) \in \mathcal{D}$.

Considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t = f(t_1, \ldots, t_k) \in \mathcal{T}_\infty$ is helpful to come to the notion of dependency pair.

Proposition 11 [HM04, Lemma 1] *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS. For all $t \in \mathcal{T}_\infty$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{T}_\infty$ such that*

$root(u) \in \mathcal{D}$, $t \xrightarrow{>\Lambda}{}^{*} \sigma(l) \xrightarrow{\Lambda} \sigma(r) \trianglerighteq u$ and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.

According to the discussion in [AGL10] and [AL11], we introduce the following:

Definition 12 ((Strongly) Minimal Innermost Non-μ-Terminating Term) *Let $\mathcal{M}_{\infty,\mu,\mathsf{i}}$ be the set of minimal innermost non-μ-terminating terms in the following sense: t belongs to $\mathcal{M}_{\infty,\mu,\mathsf{i}}$ if t is not innermost μ-terminating and every strict μ-replacing subterm s of t (i.e., $t \rhd_\mu s$) is innermost μ-terminating. Let $\mathcal{T}_{\infty,\mu,\mathsf{i}}$ be the set of strongly minimal innermost non-μ-terminating terms in the following sense: t belongs to $\mathcal{T}_{\infty,\mu,\mathsf{i}}$ if t is not innermost μ-terminating and every strict subterm u (i.e., $t \rhd u$) is innermost μ-terminating. It is obvious that $root(t) \in \mathcal{D}$ for all $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$ and $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*

Note that $\mathcal{T}_{\infty,\mu,\mathsf{i}} \subseteq \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Before starting our discussion about minimal innermost non-μ-terminating terms, we introduce some auxiliary results about innermost μ-terminating terms (see [AGL10, Lemmata 1,2,3,4]).

Proposition 13 *Let $\mathcal{R} = (\Sigma, R)$ be a TRS, $\mu \in M_\Sigma$, and $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$.*

1. *If s is innermost μ-terminating and $s \trianglerighteq_\mu t$ or $s \hookrightarrow^{*}_{\mathcal{R},\mu,\mathsf{i}} t$ then t is innermost μ-terminating.*
2. *If s is not innermost μ-terminating, then there is a subterm t of s ($s \trianglerighteq t$) such that $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$. Furthermore, there is a μ-replacing subterm t of s ($s \trianglerighteq_\mu t$) such that $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*
3. *If $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, $t \overset{>\Lambda}{\hookrightarrow}{}^{*}_{\mathsf{i}} u$ and u is not innermost μ-terminating, then $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*

The following result is the innermost context-sensitive version of Lemma 1 in [HM04] (i.e., Proposition 11), which uses the results in Proposition 13. Proposition 14 below establishes that, given a minimal innermost non-μ-terminating term $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there are only two ways for an infinite innermost μ-rewrite sequence to proceed. The first one is by using 'visible' parts of the rules that correspond to μ-replacing nonvariable subterms in the right-hand sides which are rooted by a defined symbol. This corresponds to the straightforward extension of the original result but taking into account the replacement restrictions. The second one is by showing up 'hidden' innermost non-μ-terminating subterms that are activated by *migrating* variables in a rule $l \to r$, i.e., variables $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)$ that are *not* μ-replacing in the left-hand side l but become μ-replacing in the right-hand side r.

Proposition 14 *[AL11] Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\Sigma$. Then for all $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there exist $l \to r \in R$, a substitution σ such that $\sigma(l)$ is argument μ-normalized and a term $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ such that $t \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^* \sigma(l) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma(r) \trianglerighteq_\mu u$ and either*

1. *there is a nonvariable μ-replacing subterm s of r, $r \trianglerighteq_\mu s$, such that $u = \sigma(s)$ and $\sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$ for all $x \in \mathcal{V}ar(s) \cap \mathcal{V}ar^\mu(l)$, or*

2. *there is $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)$ such that $\sigma(x) \trianglerighteq_\mu u$, that is, $\sigma(x) = C[u]_p$ for some context $C[]_p$ with $p \in Pos^\mu(C[]_p)$.*

Proposition 14 entails the following result, which establishes some properties of infinite sequences starting from minimal innermost non-μ-terminating terms.

Corollary 15 *[AL11] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. For all $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there is an infinite sequence*

$$t \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^* \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^* \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^* \cdots$$

where, for all $i \geq 1$, $l_i \to r_i \in R$ are rewrite rules, σ_i are substitutions, $\sigma_i(l_i)$ is argument μ-normalized, and terms $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ are minimal innermost non-μ-terminating terms such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable subterm s_i of r_i such that $r_i \trianglerighteq_\mu s_i$ and $\sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$ for all $x \in \mathcal{V}ar(s_i) \cap \mathcal{V}ar^\mu(l_i)$, or*

2. *$\sigma_i(x_i) \trianglerighteq_\mu t_i$, i.e., $\sigma_i(x_i) = C[t_i]_{p_i}$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$ and context $C[\,]_{p_i}$ with $p_i \in Pos^\mu(C[\,]_{p_i})$.*

Now we address item 2 of Proposition 14. To analyze in depth infinite sequences starting from minimal innermost non-μ-terminating terms, we need to go inside the instantiation of the migrating variable, $\sigma(x)$. Since in (innermost) context-sensitive rewriting, function calls can be delayed, terms that are (innermost) μ-terminating can generate future (innermost) non-μ-terminating subterms. By Proposition 13, we know that innermost μ-termination is preserved under μ-rewritings and extraction of μ-replacing subterms. Therefore, these innermost non-μ-terminating subterms introduced by applying innermost μ-rewriting steps can only occur at frozen positions in the reducts. This is captured by the notion of *hidden term*.

Definition 16 (Hidden Term [AGL10]) *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. We say that $t \in \mathcal{T}(\Sigma, \mathcal{X}) - \mathcal{X}$ is a* hidden term *if there is a rule $l \to r \in R$ such that $r \rhd_{\not\mu} t$. Let $\mathcal{HT}(\mathcal{R}, \mu)$ (or just $\mathcal{HT}$, if $\mathcal{R}$ and μ are clear from the context) be the set of all hidden terms in $(\mathcal{R}, \mu)$. We say that $f \in \Sigma$ is a* hidden symbol *if it occurs in a hidden term. Let $\mathcal{H}(\mathcal{R}, \mu)$ (or just $\mathcal{H}$) be the set of all hidden symbols in $(\mathcal{R}, \mu)$. We also use*

$$\mathcal{DHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT} \mid root(t) \in \mathcal{D}\}$$

for the set of hidden terms which are rooted by a defined *symbol.*

Example 17

In Example 7, the hidden terms are $\mathtt{from}(\mathtt{s}(x))$, $\mathtt{s}(x)$, $\mathtt{filter}(\mathtt{s}(\mathtt{s}(x)), z)$, $\mathtt{s}(\mathtt{s}(x))$, $y{:}\mathtt{filter}(x, \mathtt{sieve}(y))$, $\mathtt{filter}(x, \mathtt{sieve}(y))$ and $\mathtt{sieve}(y)$. The hidden symbols are `from`, `s`, `filter`, `(:)` and `sieve`. Finally, $\mathcal{DHT}(\mathcal{R}, \mu) = \{\mathtt{from}(\mathtt{s}(x)), \mathtt{filter}(\mathtt{s}(\mathtt{s}(x)), z), \mathtt{filter}(x, \mathtt{sieve}(y)), \mathtt{sieve}(y)\}$.

Innermost non-μ-terminating terms at frozen positions can be activated by some specific contexts. In Proposition 14-(2), the intended role of hidden terms in the binding of the migrating variable $\sigma(x) = C[u]_p$ is that there is always a hidden term u' such that $\theta(u') = u$ for some substitution θ. This context can only be composed by symbols f contained in hidden terms $f(\ldots, r_i, \ldots)$ such that $r' \rhd_{\not\mu} f(\ldots, r_i, \ldots) \trianglerighteq_\mu r_i$ for a rule $l' \to r' \in \mathcal{R}$ satisfying that:

- r_i is a nonvariable term and $\sigma(r_i) = u$, or
- r_i is a variable at a frozen position in both, l and r.

These symbols conform what is called as *hiding context*.

Definition 18 (Hiding Context [GL10]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. A function symbol f* hides position $i \in \mu(f)$ *in the rule $l \to r \in \mathcal{R}$ if $r \rhd_{\not\mu} f(r_1, \ldots, r_n)$ for some terms $r_1, \ldots, r_n$, and r_i contains a μ-replacing defined symbol (i.e., $\mathcal{P}os^\mu_\mathcal{D}(r_i) \neq \varnothing$) or a variable $x \in (\mathcal{V}ar^{\not\mu}(l) \cap \mathcal{V}ar^{\not\mu}(r)) \setminus (\mathcal{V}ar^\mu(l) \cup \mathcal{V}ar^\mu(r))$ which is μ-replacing in r_i (i.e., $x \in \mathcal{V}ar^\mu(r_i)$). We say that f hides position i in $\mathcal{R}$ if there is a rule $l \to r$ such that f hides position i in $l \to r$. A context $C[\Box]$ is* hiding *if*

1. *$C[\Box] = \Box$, or*
2. *$C[\Box] = f(t_1, \ldots, t_{i-1}, C'[\Box], t_{i+1}, \ldots, t_k)$, where f hides position i and $C'[\Box]$ is a hiding context.*

Since we are dealing with innermost μ-rewriting and all μ-replacing variables of the instantiated left-hand sides of the rules applied in an innermost μ-rewrite sequence are in μ-normal form, they cannot start any reduction even if they come from a frozen position on the right-hand side. In [GL10], this refinement also was done since, in a μ-rewrite sequence, these variables could start a reduction; however, due to minimality, these reductions would be finite.

In the following, we consider a function $\textsc{Ren}^\mu$ [AGL06, AGL10] that *independently* renames all *occurrences* of μ-replacing variables within a term t by using fresh variables that are not in $\mathcal{V}ar(t)$. Note that $\textsc{Ren}^\mu(t)$ keeps variables at non-μ-replacing positions untouched.

Proposition 19 *[AGL10] Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\Sigma$. Let $t \in \mathcal{T}(\Sigma, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution. If $\sigma(t) \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma(l)$ for some (renamed) rule $l \to r \in R$, then $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

Corollary 20 *[AGL10] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. Let $t \in \mathcal{T}(\Sigma, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution such that $\sigma(t) \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Then, $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

In the following, we write $\textsc{Narr}^\mu(t)$ [AGL10] to indicate that t is μ-narrowable (w.r.t. the intended TRS $\mathcal{R}$). We also let

$$\mathcal{NHT}(\mathcal{R}, \mu) = \{t \in \mathcal{DHT} \mid \textsc{Narr}^\mu(\textsc{Ren}^\mu(t))\}$$

be the set of *hidden terms* which are rooted by a *defined* symbol, and that, after applying $\textsc{Ren}^\mu$, become *μ-narrowable*. These notions are used and combined to model infinite innermost μ-rewrite sequences starting from strongly minimal innermost non-μ-terminating terms. See Example 25 for an example of use.

As a consequence of the previous results, we have the following.

Theorem 21 (Minimal Innermost μ-Sequence [AL11]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. For all $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$, there is an infinite sequence*

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \cdots$$

where, for all $i \geq 1$, $l_i \to r_i \in R$, σ_i is a substitution, $\sigma_i(l_i)$ is argument μ-normalized, and $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ is a minimal innermost non-μ-terminating term such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable term s_i such that $r_i \trianglerighteq_\mu s_i$, or*

2. $\sigma_i(x_i) = \theta_i(C_i[t'_i])$ *and* $t_i = \theta_i(t'_i)$ *for some variable* $x_i \in \mathcal{V}ar^{\mu}(r_i) \setminus \mathcal{V}ar^{\mu}(l_i)$, $t'_i \in \mathcal{NHT}(\mathcal{R}, \mu)$, *hiding context* $C_i[\Box]$, *and substitution* θ_i.

In Chapter 4, we explain the relevance of these results and how to use them to define an appropriate notion of innermost context-sensitive dependency pairs and of innermost μ-chain.

4

Innermost Context-Sensitive Dependency Pairs and Chains

According to Theorem 21, an infinite minimal innermost μ-rewriting sequence whose starting term t is strongly minimal has the following form:

$$t = t_0 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \cdots$$

where t_i are minimal innermost non-μ-terminating terms, for all $i \geq 1$. Then, we proceed by first performing some innermost μ-rewriting steps *below the root* of t_{i-1} to obtain a term $\sigma_i(l_i)$ (i.e., $t_{i-1} \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \sigma_i(l_i)$) such that $\sigma_i(l_i)$ is argument μ-normalized and then applying a rule $l_i \to r_i$ at the *topmost* position for some matching substitution σ_i. The application of such a rule either

1. *introduces* a new minimal innermost non-μ-terminating subterm $t_i = \sigma_i(s_i)$, where s_i is a μ-replacing nonvariable subterm of r_i which is rooted by a defined symbol (i.e., $r_i \trianglerighteq_\mu s_i$ and $\mathsf{root}(s_i) \in \mathcal{D}$), or

2. *takes* a minimal innermost non-μ-terminating and non-μ-replacing subterm t_i from $\sigma_i(l_i)$ (i.e., $\sigma_i(l_i) \rhd_{\not\mu} t_i$) and

   (a) brings it up to an *active* position by means of the binding $\sigma_i(x_i)$ for some *migrating variable* x_i in $l_i \to r_i$, $\sigma(x_i) = \theta(C_i[t'_i])$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$ and a context $C_i[\Box]$ with a μ-replacing hole.

   (b) At this point, we know that $\sigma(x_i) = \theta(C_i[t'_i])$, where t'_i is rooted by a defined symbol due to $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Furthermore, t_i is an instance of the μ-narrowable hidden term $t'_i \in \mathcal{NHT}$ (i.e. $t_i = \theta(t'_i)$) and $C_i[\Box]$ is an instance of a hiding context.

Afterwards, further *inner* μ-rewritings on t_i lead to match the left-hand-side l_{i+1} of a new rule $l_{i+1} \to r_{i+1}$, i.e., $t_i \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma_{i+1}(l_{i+1})$ for some substitution, and everything starts again. This process is abstracted in the definition of *innermost context-sensitive dependency pairs* and *innermost context-sensitive dependency chain*.

Definition 22 (Innermost Context-Sensitive Dependency Pairs [AL11]) *Let $\mathcal{R} = (\Sigma, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\Sigma$. We define $\mathsf{iDP}(\mathcal{R}, \mu) = \mathsf{iDP}_\Sigma(\mathcal{R}, \mu) \cup \mathsf{iDP}_\mathcal{X}(\mathcal{R}, \mu)$ to be the set of* innermost context-sensitive *dependency pairs (ICSDPs) where:*

$$\begin{array}{lcl} \mathsf{iDP}_\Sigma(\mathcal{R}, \mu) & = & \{l^\sharp \to s^\sharp \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s, \textsc{Narr}^\mu(\textsc{Ren}^\mu(s))\} \\ \mathsf{iDP}_\mathcal{X}(\mathcal{R}, \mu) & = & \{l^\sharp \to x \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)\} \end{array}$$

We extend $\mu \in M_\Sigma$ into $\mu^\sharp \in M_{\Sigma \cup \mathcal{D}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \Sigma$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$.

As in [HM04] (which follows Dershowitz's proposal in [Der04]), we require that subterms s of the right-hand sides r of the rules $l \to r$, which are used to build the ICSDPs $l^\sharp \to s^\sharp$, not be μ-replacing subterms of the left-hand side (i.e., $l \not\trianglerighteq_\mu s$). Also, as in [LM08], we require 'μ-narrowability' of $\textsc{Ren}^\mu(s)$: $\textsc{Narr}^\mu(\textsc{Ren}^\mu(s))$. This condition removes pairs that cannot generate infinite sequences. The ICSDPs $u \to v \in \mathsf{iDP}_\mathcal{X}(\mathcal{R}, \mu)$ in Definition 22, consisting of collapsing rules only, are called the *collapsing* ICSDPs. A rule $l \to r$ of a TRS $\mathcal{R}$ is *μ-conservative* if $\mathcal{V}ar^\mu(r) \subseteq \mathcal{V}ar^\mu(l)$, i.e., it does not contain migrating variables; $\mathcal{R}$ is *μ-conservative* if all its rules are (see [Luc96, Luc06]).

Clearly, if $\mathcal{R}$ is μ-conservative, then $\mathsf{iDP}(\mathcal{R}, \mu) = \mathsf{iDP}_\Sigma(\mathcal{R}, \mu)$. Therefore, in order to deal with μ-conservative TRSs $\mathcal{R}$ we only need to consider the 'classical' dependency pairs in $\mathsf{iDP}_\Sigma(\mathcal{R}, \mu)$ (having into account the replacement restrictions). If the TRS $\mathcal{R}$ contains non-μ-conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

Example 23

Consider Example 7. The set $\mathsf{iDP}(\mathcal{R}, \mu)$ consists of the following pairs:

$$\begin{array}{rcll} \mathtt{IF}(\mathtt{true}, x, y) & \to & x & (4.1) \\ \mathtt{IF}(\mathtt{true}, x, y) & \to & y & (4.2) \\ \mathtt{PRIMES} & \to & \mathtt{FROM}(\mathtt{s}(\mathtt{s}(0))) & (4.3) \\ \mathtt{PRIMES} & \to & \mathtt{SIEVE}(\mathtt{from}(\mathtt{s}(\mathtt{s}(0)))) & (4.4) \\ \mathtt{TAIL}(x\mathtt{:}y) & \to & y & (4.5) \end{array}$$

Note that pairs (4.3) and (4.4) represent the possible *direct* 'recursive' calls in the very same sense of original DPs whereas pairs (4.1), (4.2) and (4.5) are collapsing pairs and represent the activation of *delayed* function calls by migrating variables.

To deal with the information corresponding to hidden terms and hiding contexts when trying to characterize innermost μ-termination with ICSDPs, we use an *unhiding TRS* $\mathsf{unh}(\mathcal{R},\mu)$. This unhiding TRS captures the situation described in Theorem 21 when managing migrating variables. According to this, we have to remove the (instance of the) hiding context $C_i[]$ to extract the delayed call t_i and then connect this delayed call, which is an instance $\theta(t'_i)$ of a hidden term t'_i with the next pair in the innermost μ-chain. We perform these two actions by using two kinds of rewrite rules:

- If $\theta(C_i[t'_i]) = \theta(f(t_1,\ldots,t_{i-1},C'_i[t'_i],t_{i+1},\ldots,t_k))$ then, since $C_i[]$ is a hiding context, f hides position i and $C'_i[]$ is a hiding context as well. Then, we can extract $\theta(C'_i[t'_i])$ from $\theta(C_i[t'_i])$ by using the following projection rule: $f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_k) \to x_i$
- Once t_i has been reached, we know that it is an instance $t_i = \theta(t'_i)$ of a nonvariable hidden term $t'_i \in \mathcal{NHT}(\mathcal{R},\mu)$ and we have to connect t_i with the next innermost context-sensitive dependency pair. Since the root of the innermost context-sensitive dependency pair is a marked symbol, we can do it by using a rule that just changes the root symbol by its marked version in the following way: $t'_i \to t'^{\sharp}_i$

Definition 24 (Unhiding TRS [GL10]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We define $\mathsf{unh}(\mathcal{R},\mu)$ as the TRS consisting of the following rules:*

1. *$f(x_1,\ldots,x_i,\ldots,x_k) \to x_i$ for all function symbols f of arity k, distinct variables $x_1,\ldots,x_k$, and $1 \le i \le k$ such that f* hides position *i in $l \to r \in R$, and*
2. *$t \to t^{\sharp}$ for every $t \in \mathcal{NHT}(\mathcal{R},\mu)$.*

Example 25

Continuing with Example 7, we have that `filter` hides position 2 and `sieve` hides position 1. Therefore, the following rules have to be added to the unhiding TRS $\mathsf{unh}(\mathcal{R},\mu)$.

$$\begin{aligned} \mathtt{filter}(x,y) &\to y \\ \mathtt{sieve}(x) &\to x \end{aligned}$$

They fit the requirements in Definition 24-(1). Moreover, $\mathcal{NHT}(\mathcal{R},\mu) = \{\mathtt{filter}(\mathtt{s}(\mathtt{s}(x)),z), \mathtt{filter}(x,\mathtt{sieve}(y)), \mathtt{sieve}(x), \mathtt{from}(\mathtt{s}(x))\}$. Therefore, we also have to add the following rules:

$$\begin{aligned} \mathtt{filter}(\mathtt{s}(\mathtt{s}(x)),z) &\to \mathtt{FILTER}(\mathtt{s}(\mathtt{s}(x)),z) \\ \mathtt{filter}(x,\mathtt{sieve}(y)) &\to \mathtt{FILTER}(x,\mathtt{sieve}(y)) \\ \mathtt{sieve}(x) &\to \mathtt{SIEVE}(x) \\ \mathtt{from}(\mathtt{s}(x)) &\to \mathtt{FROM}(\mathtt{s}(x)) \end{aligned}$$

These rules refer to Definition 24-(2) and complete the unhiding TRS $\mathsf{unh}(\mathcal{R},\mu)$.

Definitions 22 and 24 lead to a suitable notion of *chain* which captures infinite minimal innermost μ-rewrite sequences according to the description in Theorem 21. In the following, given a CS-TRS $\mathcal{S}$, we let $\mathcal{S}_{\triangleright_\mu}$ be the rules from $\mathcal{S}$ of the form $s \to t \in \mathcal{S}$ and $s \triangleright_\mu t$ (cf. Definition 24-(1)); and $\mathcal{S}_\sharp = \mathcal{S} \setminus \mathcal{S}_{\triangleright_\mu}$(cf. Definition 24-(2)).

In innermost *CSR*, we only perform reduction steps on *innermost* μ-*replacing redexes*. Therefore, we have to restrict the definition of chains in order to obtain an appropriate notion corresponding to innermost *CSR*, which, obviously, is an adaptation of the one for standard *CSR* (see [AGL10, GL10]). Regarding innermost reductions, arguments of a redex should be in *normal form* before the redex is contracted; regarding *CSR*, the redex to be contracted has to be in a μ-*replacing* position. As in the DP framework where the procedence of *pairs* does not matter, we instead think of another TRS $\mathcal{P}$ that is used together with $\mathcal{R}$ to build the chains. Once this more abstract notion of chain is introduced, it can be particularized for use with ICSDPs, by just taking $\mathcal{P} = \mathsf{iDP}(\mathcal{R},\mu)$.

Definition 26 ((Minimal) Innermost μ-Chain of Pairs [AL11]) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. An innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution σ satisfying that, for all $i \geq 1$, $\sigma(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$ and :*

1. *if $v_i \notin \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i) = t_i \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathbf{i}} \sigma(u_{i+1})$, and*

2. *if* $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$, *then* $\sigma(v_i) \xhookrightarrow{\Lambda}{}^{*}_{\mathcal{S}_{\triangleright_\mu},\mu} \circ \xhookrightarrow{\Lambda}_{\mathcal{S}_\sharp,\mu} t_i \xhookrightarrow{!}_{\mathcal{R},\mu,\mathrm{i}} \sigma(u_{i+1})$.

An innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$*-chain is called* minimal *if for all* $i \geq 1$, t_i *is innermost* $(\mathcal{R},\mu)$*-terminating.*

Note that the condition $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$ in Definition 26 implies that v_i is a variable. Furthermore, since each $u_i \to v_i \in \mathcal{P}$ is a rewrite rule (i.e., $\mathcal{V}ar(v_i) \subseteq \mathcal{V}ar(u_i)$), v_i is a *migrating variable* in the rule $u_i \to v_i$.

An essential property of the dependency pair method is that it provides a *characterization* of termination of TRSs $\mathcal{R}$ as the absence of infinite (minimal) *chains of dependency pairs* [AG00, GTSF06]. As we prove here, this is also true for innermost *CSR*. The notions of ICSDP and unhiding TRS give rise to a sound and complete characterization of innermost termination of *CSR* where the set $\mathcal{P}$ corresponds to the ICSDPs, the set $\mathcal{R}$ to the rules of the system and the set $\mathcal{S}$ corresponds to the unhiding TRS.

Theorem 27 (Characterization of Innermost μ-Termination [AL11]) *Let* $\mathcal{R}$ *be a TRS and* $\mu \in M_{\mathcal{R}}$. *Let* $\mathcal{P} = \mathsf{iDP}(\mathcal{R},\mu)$ *and* $\mathcal{S} = \mathsf{unh}(\mathcal{R},\mu)$. *Then,* $\mathcal{R}$ *is innermost* μ*-terminating if and only if there is no infinite minimal* $(\mathcal{P},\mathcal{R},\mathcal{S},\mu^\sharp,\mathbf{i})$*-chain.*

Example 28

Consider the following TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c}(\mathtt{b}) \\ \mathtt{f}(\mathtt{c}(x),x) & \to & \mathtt{f}(x,x) \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ and $\mu(\mathtt{c}) = \varnothing$. There is only one ICSDP:

$$\mathtt{F}(\mathtt{c}(x),x) \to \mathtt{F}(x,x)$$

Since $\mu^\sharp(\mathtt{F}) = \{1,2\}$, if a substitution σ satisfies $\sigma(\mathtt{F}(\mathtt{c}(x),x)) \in \mathsf{NF}_\mu(\mathcal{R})$, then $\sigma(x) = s$ is in μ-normal form. Assume that the dependency pair is part of an innermost μ-chain. Since there is no way to μ-rewrite $F(s,s)$, there must be $F(s,s) = F(c(t),t)$ for some term t, which means that $s = t$ and $c(t) = s$, i.e., $t = c(t)$, which is not possible. Thus, there is no infinite innermost chain of ICSDPs for $\mathcal{R}$, which is proved innermost terminating by Theorem 27.

Of course, ad-hoc reasonings like in Example 28 do not lead to automation. In the following chapters we discuss how to prove termination of innermost *CSR* automatically.

5

Innermost Context-Sensitive Dependency Pair Framework

The dependency pair approach emphasizes (at least theoretically) a 'linear' (although somehow *modular*, see [GAO02]) procedure for proving termination. In this sense, the treatment of strongly connected components of the graph (SCCs) instead of cycles, as suggested by Hirokawa and Middeldorp [HM04, HM05], brought an important improvement in its practical use because it provides a way to make the proofs more incremental without running out of the basic DP approach. In the DP approach, dependency pairs are considered as components of the chains (or cycles). Since they only make sense when an underlying TRS is given as the source of the dependency pairs, transforming DPs is possible (the *narrowing* transformation is already described in [AG00]) but only as a final step because, afterwards, they are not dependency pairs of the original TRS anymore. The dependency pair framework solves these problems in a clean way, leading to a more powerful mechanization of termination proofs. First, we recall the definition of minimal μ-chain to define a more general notion of CS problem.

Definition 29 ((Minimal) μ-Chain of Pairs [GL10]) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. A $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution σ satisfying that, for all $i \geq 1$,*

1. *if $v_i \notin \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$, then $\sigma(v_i) = t_i \hookrightarrow^{*}_{\mathcal{R},\mu} \sigma(u_{i+1})$, and*
2. *if $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$, then $\sigma(v_i) \xhookrightarrow{\Lambda}{}^{*}_{\mathcal{S}_{\rhd\mu},\mu} \circ \xhookrightarrow{\Lambda}_{\mathcal{S}_{\sharp},\mu} t_i \hookrightarrow^{*}_{\mathcal{R},\mu} \sigma(u_{i+1})$.*

A $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain is called minimal *if for all $i \geq 1$, t_i is $(\mathcal{R},\mu)$-terminating.*

Definition 30 (CS Problem) *A CS problem τ is a tuple $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$, where $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ are TRSs, $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$, and $e \in \{\mathbf{t}, \mathbf{i}\}$ is a flag that stands for* **t**ermination *or* **i**nnermost termination *of* CSR. *The* CS problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$ *is* finite *if there is no infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$-chain. The* CS problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$ *is* infinite *if $\mathcal{R}$ is non-μ-terminating (for $e = \mathbf{t}$) or innermost non-μ-terminating (for $e = \mathbf{i}$) or there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$-chain.*

Definition 31 (CS Processor) *A CS processor* Proc *is a mapping from CS problems into sets of CS problems. Alternatively, it can also return* "no". *A CS processor* Proc *is*

- sound *if for all CS problems τ, τ is finite whenever $\mathsf{Proc}(\tau) \neq \mathsf{no}$ and $\forall \tau' \in \mathsf{Proc}(\tau)$, τ' is finite.*
- complete *if for all CS problems τ, τ is infinite whenever $\mathsf{Proc}(\tau) = \mathsf{no}$ or $\exists \tau' \in \mathsf{Proc}(\tau)$ such that τ' is infinite.*

Now we have the following result which extends the framework in [GL10] to innermost *CSR*.

Theorem 32 (CSDP Framework [AL11]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We construct a tree whose nodes are labeled with CS problems or* "yes" *or* "no", *and whose root is labeled with $(\mathcal{P}, \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp, e)$, where $\mathcal{P} = \mathsf{DP}(\mathcal{R}, \mu)$ if $e = \mathbf{t}$ and $\mathcal{P} = \mathsf{iDP}(\mathcal{R}, \mu)$ if $e = \mathbf{i}$. For every inner node labeled with τ, there is a sound processor* Proc *satisfying one of the following conditions:*

1. *$\mathsf{Proc}(\tau) = \mathsf{no}$ and the node has just one child, labeled with* "no".
2. *$\mathsf{Proc}(\tau) = \varnothing$ and the node has just one child, labeled with* "yes".
3. *$\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$, and the children of the node are labeled with the CS problems in $\mathsf{Proc}(\tau)$.*

If all leaves of the tree are labeled with "yes", *then $\mathcal{R}$ is (innermost) μ-terminating if $e = \mathbf{t}$ (resp. $\mathbf{i}$). Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is not (innermost) μ-terminating.*

Of course, the termination problems that we treat here are undecidable (in general), thus "don't know" answers can also be generated (for instance, by a timeout system that interrupts the usually complex search processes that are involved in the proofs).

Remark 33 According to Theorem 27, we can say now that a TRS $\mathcal{R}$ is innermost μ-terminating if and only if the CS problem $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$ is finite. ■

Example 34

Continuing with Example 7, in order to prove innermost termination of $(\mathcal{R},\mu)$, we start with the following CS problem:

$$\tau = (\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$$

where $\mathsf{iDP}(\mathcal{R},\mu)$ is obtained in Example 23 and $\mathsf{unh}(\mathcal{R},\mu)$ is obtained in Example 25.

In the following chapters we describe a number of sound and (mostly) complete CS processors for proving termination of innermost *CSR*. First, we are going to comment on some interesting points that relate innermost termination of *CSR* to μ-termination.

5.1 Innermost Termination and Termination of *CSR*

Our definition of ICSDPs coincides (in the most important points) with the standard one for proving termination of innermost rewriting if no replacement restrictions are considered (equivalently, if the top replacement map $\mu_\top(f) = \{1,\ldots,ar(f)\}$ for all $f \in \Sigma$ is used). Of course, when proving μ-termination of a TRS, we are also proving innermost μ-termination (the converse does not hold). For this reason, although many standard techniques that we have developed for proving termination of *CSR* are not mentioned here (see [AGL10]), they can also be used for proving innermost μ-termination (as in full rewriting). However, the other direction is more interesting, since proving innermost termination usually offers simpler proofs. Something similar happens with *CSR*.

5.1.1 Switching to Innermost Termination of *CSR*

In standard rewriting, Gramlich showed that termination and innermost termination coincide for locally confluent overlay TRSs $\mathcal{R}$ [Gra95, Theorem 3.23]. Thus, his result allows us to prove termination of such TRSs $\mathcal{R}$ by proving innermost termination of $\mathcal{R}$. Although local confluence is undecidable, every nonoverlapping rewrite system is also a locally confluent overlay system; therefore, this approximation is commonly adopted. However, for context-sensitive

rewriting this is not enough. This fact was noticed by Lucas [Luc01c], showing the following example:

Example 35

Consider the following TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{f}(x,x) & \to & \mathtt{b} \\ \mathtt{f}(x,g(x)) & \to & \mathtt{f}(x,x) \\ \mathtt{c} & \to & \mathtt{g}(c) \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ and $\mu(\mathtt{g}) = \varnothing$. This system is nonoverlapping and innermost μ-terminating, but not μ-terminating since $\mathtt{f}(\mathtt{c},\mathtt{c}) \hookrightarrow_\mu \underline{\mathtt{f}(\mathtt{c},\mathtt{g}(\mathtt{c}))} \hookrightarrow_\mu \mathtt{f}(\mathtt{c},\mathtt{c}) \hookrightarrow_\mu \cdots$

Later, in [GL02b, GM02a] it is proved that the equivalence between termination of innermost *CSR* and termination of *CSR* holds in some interesting cases. Thanks to this, the following result was formulated:

Theorem 36 *[GM02a] Let $\mathcal{R} = (\Sigma, R)$ be an orthogonal TRS and $\mu \in M_\Sigma$. $\mathcal{R}$ is μ-terminating if and only if it is innermost μ-terminating.*

If $\mathcal{R}$ is orthogonal, the innermost μ-rewriting relation is locally confluent and therefore, every innermost μ-terminating term has a unique normal form. A similar result can be found in [GL02b]. First, we need the following definition:

Definition 37 *[Luc98, Definition 5] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\mathcal{R}$. $\mathcal{R}$ has* left-homogeneous replacing variables *(LHRV for short) if, for every μ-replacing variable x in the left-hand side l of a rule $l \to r \in R$, all occurrences of x are replacing in both l and r.*

Theorem 38 *[GL02b, Theorem 7] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\mathcal{R}$ such that $\mathcal{R}$ is a locally confluent overlay system satisfying LHRV. If $\mathcal{R}$ is innermost μ-terminating, then it is also μ-terminating.*

So, whenever it is possible, we switch to innermost μ-termination since proofs are often easier due to the fact that, when considering an innermost rewriting step, we know that every possible subterm of our redex is in normal form with respect to our rewriting relation. For instance, this is shown when estimating the graph.

On the other hand, we have developed many processors for proving termination of *CSR* [AGL10, GL10] and it is also interesting to use them in proofs of innermost termination of *CSR*.

Theorem 39 (Commuting Processors [AL11]) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem such that*

1. *$(\mathcal{R} \cup \mathcal{P} \cup \mathcal{S})$ is nonoverlapping and satisfies LHRV, or*

2. *$(\mathcal{R} \cup \mathcal{P} \cup \mathcal{S})$ is orthogonal.*

Then, the processors $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$ and $\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}$ given by

$$\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t}) = \begin{cases} \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{if (1) or (2)} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{otherwise} \end{cases}$$

$$\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{if (1) or (2)} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

are sound and $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$ is also complete.

Soundness of $\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}$ needs to impose the requirements about equivalence between innermost μ-termination and μ-termination since we are dealing with minimal chains. Obviously, it is always possible to prove innermost μ-termination of a TRS by proving μ-termination without taking into account any additional condition but this cannot be done when managing minimality.

Now we show that our framework coincides with the standard one for proving full (innermost) termination when no replacement map is considered.

5.1.2 ICSDPs and IDPs

Given a TRS $\mathcal{R}$ and a replacement map μ, if no replacement restrictions are imposed, i.e., $\mu(f) = \{1, \ldots, ar(f)\}$ for all $f \in \Sigma$, then no collapsing pair is possible, and we would have $\mathsf{iDP}(\mathcal{R}, \mu) = \mathsf{iDP}_\Sigma(\mathcal{R}, \mu)$.

Regarding the pairs in $\mathsf{iDP}_\Sigma(\mathcal{R}, \mu)$, Definition 22 differs from the standard definition of dependency pair (e.g., [AG00, GTSF06]) in three additional requirements:

1. As in [HM04], which follows Dershowitz's proposal in [Der04], we require that subterms s of the right-hand sides r of the rules $l \to r$ which are considered to build the dependency pairs $l^\sharp \to s^\sharp$ are not subterms of the left-hand side (i.e., $l \not\trianglerighteq_\mu s$).

2. As in [LM08], we require 'narrowability' of the (appropriately renamed) term s: $\textsc{Narr}^\mu(\textsc{Ren}^\mu(s))$.

3. We explicitly require that the left-hand side l of a rule $l \to r$ is argument μ-normalized when considering the dependency pair $l^\sharp \to s^\sharp$ (or $l^\sharp \to x$), that is, $l^\sharp \in \mathsf{NF}_\mu(\mathcal{R})$. In the standard definition, this is exploited when building the graph but, in our definition, we avoid the introduction of spurious pairs from the beginning.

Except for these provisos, we could say that Definition 22 boils down to the definition of dependency pair when no replacement restrictions are imposed.

Regarding the definition of (minimal) chain (Definition 26), the correspondence is exact: if μ imposes no replacement restriction, then $\to_{\mathcal{R},\mathsf{i}} = \hookrightarrow_{\mathcal{R},\mu,\mathsf{i}}$ and our definition coincides with the standard one (see, e.g., [GTSF06, Definition 3]): again, since all variables are μ-replacing now, item (2) in Definition 26 does not apply. Due to the absence of replacement restrictions, we have $\mathcal{V}ar^\mu(u) = \mathcal{V}ar(u)$, hence $\mathcal{V}ar(u) \setminus \mathcal{V}ar^\mu(u) = \varnothing$ for all $u \to v \in \mathcal{P}$. Then, the condition $v \notin \mathcal{V}ar(u) \setminus \mathcal{V}ar^\mu(u)$ vacuously holds and all pairs in $\mathcal{P}$ satisfy item (1) of Definition 26.

6

ICS Processors

The following proposition establishes some important 'basic' cases of (absence of) infinite innermost context-sensitive chains of pairs. With slight differences they were presented in [AGL10] for *CSR*. In the following, given a CS-TRS $(\mathcal{P},\mu)$ where $\mathcal{P}=(\mathcal{G},P)$, we let $\mathcal{P}_\mathcal{X}$ be the pairs $u\to v\in\mathcal{P}$ such that $v\in\mathcal{V}ar(u)\setminus\mathcal{V}ar^\mu(u)$; and $\mathcal{P}_\mathcal{G}=\mathcal{P}\setminus\mathcal{P}_\mathcal{X}$.

Proposition 40 *[AL11] Let $\mathcal{R}=(\Sigma,R)$, $\mathcal{P}=(\mathcal{G},P)$, and $\mathcal{S}=(\mathcal{H},S)$ be TRSs and $\mu\in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$.*

1. *If $P=\varnothing$, then there is no $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*
2. *If $R=\varnothing$, then there is no infinite $(\mathcal{P}_\mathcal{X},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*
3. *Let $u\to v\in\mathcal{P}_\mathcal{G}$ be such that $v'=\theta(u)$ for some substitution θ such that $\theta(u)\in\mathsf{NF}_\mu(\mathcal{R})$ and renamed version v' of v. Then, there is an infinite innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*

According to Proposition 40, for some specific CS problems it is easy to say whether they are finite or not.

Theorem 41 (Basic Innermost CS Processors) *Let $\mathcal{R}=(\Sigma,R)$, $\mathcal{P}=(\mathcal{G},P)$, and $\mathcal{S}=(\mathcal{H},S)$ be TRSs and $\mu\in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. Then, the processors Proc_{Fin} and Proc_{Inf} given by*

$$\mathsf{Proc}_{Fin}(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})=\begin{cases}\varnothing & \text{if } P=\varnothing\vee(R=\varnothing\wedge\mathcal{P}=\mathcal{P}_\mathcal{X});\text{ and}\\ \{(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})\} & \text{otherwise}\end{cases}$$

$$\mathsf{Proc}_{Inf}(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})=\begin{cases}\mathsf{no} & \text{if } v=\theta(u)\text{ and }\theta(u)\in\mathsf{NF}_\mu(\mathcal{R})\\ & \text{for some } u\to v\in\mathcal{P}_\mathcal{G}\text{ and substitution }\theta;\text{ and}\\ \{(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})\} & \text{otherwise}\end{cases}$$

are sound and complete.

The CS problems in Theorem 41 provide the necessary *base* cases for our proofs of innermost termination of *CSR*. In the following sections we are going to show some powerful specific techniques to deal with proofs of innermost termination of *CSR*.

6.1 Innermost Context-Sensitive Dependency Graph

In general, an infinite sequence $S = a_1, a_2, \ldots, a_n, \ldots$ of objects a_i belonging to a set A can be represented as a path in a *graph* G whose nodes are the objects in A, and whose arcs among them are appropriately established to represent S (in particular, an arc from a_i to a_{i+1} should be established if we want to be able to capture the sequence above). Actually, if A is *finite*, then the infinite sequence S defines at least one *cycle* in G: since there is a finite number of different objects $a_i \in A$ in S, there is an infinite tail $S' = a_m, a_{m+1}, \ldots$ of S where *all objects* a_i *occur infinitely many times* for all $i \geq m$. This clearly corresponds to a cycle in G.

Given a CS problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, e)$, the analysis of the cycles in the graph build from pairs in $\mathcal{P}$ is useful to investigate the existence of infinite (minimal) chains of pairs.

Definition 42 (Innermost Context-Sensitive Graph of Pairs [AL11]) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. The* innermost context-sensitive graph $\mathsf{IG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ *has $\mathcal{P}$ as the set of nodes. Given $u \to v, u' \to v' \in \mathcal{P}$, there is an arc from $u \to v$ to $u' \to v'$ if $u \to v, u' \to v'$ is a minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain for some substitution σ.*

In termination proofs, we are concerned with the analysis of *SCCs* . An SCC in a graph is a *maximal cycle*, i.e., a cycle which is not contained in any other cycle. The following result justifies the use of SCCs for proving the absence of infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chains.

Theorem 43 (SCC Processor [GL10]) *Let $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. Then, the processor* Proc_{SCC} *given by*

$$\mathsf{Proc}_{SCC}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}_\mathcal{Q}, \mu, \mathbf{i}) \mid \mathcal{Q} \text{ contains the pairs of an SCC in } \mathsf{IG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\}$$

(where $\mathcal{S}_\mathcal{Q}$ are the rules from $\mathcal{S}$ involving a possible $(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain) is sound and complete.

As a consequence of this theorem, we can *separately* work with the strongly connected components of $\mathsf{IG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$, disregarding other parts of the graph.

Now we can use these notions to introduce the innermost context-sensitive dependency graph, i.e., the graph whose nodes instead of being arbitrary pairs are the ICSDPs ($\mathcal{P} = \mathsf{iDP}(\mathcal{R}, \mu)$).

Definition 44 (Innermost Context-Sensitive Dependency Graph [AL11]) *Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_\Sigma$. The* innermost context-sensitive dependency graph *(ICS-DG) associated to $\mathcal{R}$ and μ is*

$$\mathsf{IDG}(\mathcal{R}, \mu) = \mathsf{IG}(\mathsf{iDP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp)$$

6.1.1 Estimating the Innermost Context-Sensitive Graph

In general, the graph of a CS problem is *not* computable. It involves reachability of two pairs using (innermost) *CSR*; as in the unrestricted case, the reachability problem for innermost *CSR* is undecidable. So, we need to use some approximation of it. In [AGL10], we have adapted to the context-sensitive setting the approximation for standard rewriting of [GTS05]. The idea is to obtain the maximal prefix context $C[\Box]$ of s (i.e., $s = C[s_1, \ldots, s_k]$ for some terms $s_1, \ldots, s_k$) that we know (without any 'look-ahead' for applicable rules) cannot be changed by any reduction starting from s. Furthermore, the above terms $s_1, \ldots, s_k$ must be rooted by defined symbols. Now, we replace those subterms s_i that are at μ-replacing positions (i.e., $s_i = s|_{p_i}$ for some $p_i \in \mathcal{P}os^\mu(s)$) by fresh variables x, and we leave the non-μ-replacing ones untouched.

Definition 45 *[AGL10] Given a TRS $\mathcal{R}$ and a replacement map μ, we let $\mathrm{TCAP}^\mu_\mathcal{R}$ be as follows:*

$$\begin{array}{rcl} \mathrm{TCAP}^\mu_\mathcal{R}(x) & = & y \ \textit{if } x \textit{ is a variable, and} \\ \mathrm{TCAP}^\mu_\mathcal{R}(f(t_1, \ldots, t_k)) & = & \begin{cases} f([t_1]^f_1, \ldots, [t_k]^f_k) & \textit{if } f([t_1]^f_1, \ldots, [t_k]^f_k) \textit{ does not unify} \\ & \textit{with } l \textit{ for any } l \to r \textit{ in } \mathcal{R} \\ y & \textit{otherwise} \end{cases} \end{array}$$

where y is intended to be a new, fresh variable that has not yet been used and given a term s, $[s]^f_i = \mathrm{TCAP}^\mu_\mathcal{R}(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that l shares no variable with $f([t_1]^f_1, \ldots, [t_k]^f_k)$ when the unification is attempted.

Function $\mathrm{TCAP}^\mu_\mathcal{R}$ is intended to provide a suitable approximation of the aforementioned $(\mathcal{R}, \mu)$-reachability problems by means of *unification*.

In contrast to standard (μ-)rewriting, in the innermost setting it is not necessary to rename multiple occurrences of variables since all variables are always instantiated to (μ-)normal forms and cannot be reduced. However, in innermost *CSR* we have to replace by fresh variables those ones that are μ-replacing in the right-hand side v of the pair, but not in the left-hand side, u, since they are not guaranteed to be μ-normalized. Moreover we need to

substitute every μ-replacing subterm with a defined root symbol by fresh variables only if the term is not equal to a μ-replacing subterm of u or it unifies with the left-hand side of some rule in $\mathcal{R}$.

We define a new version of the function, $\mathrm{iTCAP}^{\mu}_{\mathcal{R},u}$, which is able to approximate the ICS graph by taking into account these particularities of innermost *CSR*.

Definition 46 *[AL11] Given a TRS $\mathcal{R}$, a replacement map μ and a term u, we let $\mathrm{iTCAP}^{\mu}_{\mathcal{R},u}$ be as follows:*

$$\mathrm{iTCAP}^{\mu}_{\mathcal{R},u}(x) = \begin{cases} y & \text{if } x \in \mathcal{X} \text{ and } x \notin \mathcal{V}ar^{\mu}(u) \\ x & \text{otherwise} \end{cases}$$

$$\mathrm{iTCAP}^{\mu}_{\mathcal{R},u}(f(t_1,\ldots,t_k)) = \begin{cases} f([t_1]^f_1,\ldots,[t_k]^f_k) & \text{if } f([t_1]^f_1,\ldots,[t_k]^f_k) \text{ does not unify with } l \text{ for any } l \to r \text{ in } \mathcal{R} \text{ or it is equal to a } \mu\text{-replacing subterm of } u \\ y & \text{otherwise} \end{cases}$$

where y is intended to be a new, fresh variable that has not yet been used and given a term s, $[s]^f_i = \mathrm{iTCAP}^{\mu}_{\mathcal{R},u}(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that l shares no variable with $f([t_1]^f_1,\ldots,[t_k]^f_k)$ when the unification is attempted.

Since when connecting in a chain collapsing pairs we deal with rules in $\mathcal{S}_\sharp$ instead of pairs in $\mathcal{P}_\mathcal{G}$, we cannot look at the left-hand sides of the pairs. Therefore, for dealing with pairs in $\mathcal{P}_\mathcal{X}$, we have to approximate their arcs in the same way as for *CSR* since we do not store information about left-hand sides of the pairs from which the hidden terms are obtained. So, we have the following:

Definition 47 (Estimated ICS-Graph of Pairs [AL11]) *Let $\mathcal{R} = (\Sigma, R)$, $\mathcal{P} = (\mathcal{G}, P)$, and $\mathcal{S} = (\mathcal{H}, S)$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. The* estimated *ICS-graph associated to $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{S}$ (denoted $\mathsf{EIG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$) has $\mathcal{P}$ as the set of nodes and arcs which connect them as follows:*

1. *There is an arc from $u \to v \in \mathcal{P}_\mathcal{G}$ to $u' \to v' \in \mathcal{P}$ if $\mathrm{iTCAP}^{\mu}_{\mathcal{R},u}(v)$ and u' unify by some mgu σ such that $\sigma(u), \sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.*
2. *There is an arc from $u \to v \in \mathcal{P}_\mathcal{X}$ to $u' \to v' \in \mathcal{P}$ if there is $s \to t \in \mathcal{S}_\sharp$ such that $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(t)$ and u' unify by some mgu σ such that $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.*

Proposition 48 (Characterization of the ICS-Graph of Pairs [AL11]) *Let $\mathcal{R} = (\Sigma, R)$, $\mathcal{P} = (\mathcal{G}, P)$, and $\mathcal{S} = (\mathcal{H}, S)$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. The* estimated *ICS-graph associated to $\mathcal{R}$, $\mathcal{P}$, and $\mathcal{S}$ (denoted $\mathsf{EIG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$) has $\mathcal{P}$ as the set of nodes and arcs which connect them as follows:*

1. *If there is an arc from* $u \to v \in \mathcal{P}_{\mathcal{G}}$ *to* $u' \to v' \in \mathcal{P}$ *and substitutions* θ *and* θ' *such that* $\theta(v) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \theta'(u')$, $\theta(u), \theta'(u') \in \mathsf{NF}_\mu(\mathcal{R})$, *then* $\mathsf{iTCAP}^\mu_{\mathcal{R},u}(v)$ *and* u' *unify by some mgu* σ *such that* $\sigma(u), \sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.

2. *If there is an arc from* $u \to v \in \mathcal{P}_{\mathcal{X}}$ *to* $u' \to v' \in \mathcal{P}$ *and there is* $s \in \mathcal{NHT}(\mathcal{R}, \mu)$ *such that* $\theta(s^\sharp) = \theta(t) = t'$ *and substitutions* θ *and* θ' *such that* $\theta(v) \overset{\Lambda}{\longrightarrow}^*_{\mathcal{S}_{\triangleright_\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} t' \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \theta'(u'), \theta'(u') \in \mathsf{NF}_\mu(\mathcal{R})$, *then there is* $s \to t \in \mathcal{S}_\sharp$ *such that* $\mathrm{TCAP}^\mu_{\mathcal{R}}(t)$ *and* u' *unify by some mgu* σ *such that* $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.

Therefore, the estimated *CS graph* $\mathsf{EIG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ *contains the ICS graph* $\mathsf{IG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$.

According to Definition 44, we would have the corresponding one for the *estimated* ICS-DG: $\mathsf{EIDG}(\mathcal{R}, \mu) = \mathsf{EIG}(\mathsf{iDP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp)$.

Since for approximating the ICS graph of a set of pairs, we use function $\mathrm{TCAP}^\mu_{\mathcal{R}}$ for connecting pairs in $\mathcal{P}_{\mathcal{X}}$ as done in the context-sensitive case, we can also use the following processor instead, which allows a better approximation of the SCCs. This is because if the SCC has no collapsing pairs, the set $\mathcal{S}$ has no sense and if it has, some pairs from $\mathcal{S}_\sharp$ can be removed: those that are not involved in the unification process. Therefore, we will always compute the SCCs by applying the following processor:

Theorem 49 (SCC Processor using $\mathrm{TCAP}^\mu_{\mathcal{R}}$ **[GL10])** *Let* $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ *be a CS problem. The CS processor* Proc_{SCC} *given by*

$$\mathsf{Proc}_{SCC}(\tau) = \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}_\mathcal{Q}, \mu, \mathbf{i}) \mid \mathcal{Q} \text{ contains the pairs of an SCC in } \mathsf{EIG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\}$$

where

- $\mathcal{S}_\mathcal{Q} = \varnothing$ *if* $\mathcal{Q}_\mathcal{X} = \varnothing$.
- $\mathcal{S}_\mathcal{Q} = \mathcal{S}_{\triangleright_\mu} \cup \{s \to t \mid s \to t \in \mathcal{S}_\sharp, \mathrm{TCAP}^\mu_{\mathcal{R}}(t)$ *and* u' *unify for some* $u' \to v' \in \mathcal{Q}$ *by some mgu* σ *such that* $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})\}$ *if* $\mathcal{Q}_\mathcal{X} \neq \varnothing$.

is sound and complete.

Example 50

Consider the following TRS $\mathcal{R}$ [Zan97, Example 4]:

$$\begin{aligned}
\mathtt{f}(x) &\to \mathtt{cons}(x, \mathtt{f}(\mathtt{g}(x))) \\
\mathtt{g}(\mathtt{0}) &\to \mathtt{s}(\mathtt{0}) \\
\mathtt{g}(\mathtt{s}(x)) &\to \mathtt{s}(\mathtt{s}(\mathtt{g}(x))) \\
\mathtt{sel}(\mathtt{0}, \mathtt{cons}(x, y)) &\to x \\
\mathtt{sel}(\mathtt{s}(x), \mathtt{cons}(y, z)) &\to \mathtt{sel}(x, z)
\end{aligned}$$

with $\mu(\mathtt{0}) = \varnothing$, $\mu(\mathtt{f}) = \mu(\mathtt{g}) = \mu(\mathtt{s}) = \mu(\mathtt{cons}) = \{1\}$, and $\mu(\mathtt{sel}) = \{1, 2\}$. Then, $\mathsf{iDP}(\mathcal{R}, \mu)$ consists of the following pairs:

$$\mathtt{G}(\mathtt{s}(x)) \to \mathtt{G}(x) \tag{6.1}$$

$$\mathtt{SEL}(\mathtt{s}(x), \mathtt{cons}(y, z)) \to \mathtt{SEL}(x, z) \tag{6.2}$$

$$\mathtt{SEL}(\mathtt{s}(x), \mathtt{cons}(y, z)) \to z \tag{6.3}$$

and the unhiding rules are:

$$\mathsf{unh}_{\triangleright_\mu}(\mathcal{R}, \mu) = \{\mathtt{f}(x) \to x\}$$

$$\mathsf{unh}_{\sharp}(\mathcal{R}, \mu) = \{\mathtt{f}(\mathtt{g}(x)) \to \mathtt{F}(g(x)), \mathtt{g}(x) \to \mathtt{G}(x)\}$$

Regarding pairs (6.1) and (6.2) in $\mathsf{iDP}_\Sigma(\mathcal{R}, \mu)$, there is an arc from (6.1) to itself and another one from (6.2) to itself. Regarding the only collapsing pair (6.3), we have $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(\mathtt{F}(\mathtt{g}(x))) = \mathtt{F}(y)$ and $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(\mathtt{G}(x)) = \mathtt{G}(y)$. Since $\mathtt{F}(y)$ does not unify with the left-hand side of any pair, and $\mathtt{G}(y)$ unifies with the left-hand side $\mathtt{G}(\mathtt{s}(\mathrm{x}))$ of (6.1) and $\mathtt{G}(\mathtt{s}(\mathrm{x}))$ is in μ-normal form, there is an arc from (6.3) to (6.1), see Figure 6.1. Thus, for the initial problem $\tau = (\mathsf{iDP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp, \mathbf{i})$ we have

$$\mathsf{Proc}_{SCC}(\tau) \quad = \quad \{(\{6.1\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}), (\{6.2\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i})\}$$

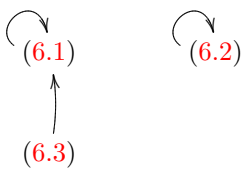


Figure 6.1: Innermost CS-dependency graph for Example 50

Example 51

Continuing with Example 7, we have started with the initial problem shown in Example 34:

$$\tau = (\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$$

If we apply the SCC processor, we get

$$\mathsf{Proc}_{SCC}(\tau) \quad = \quad \{(\varnothing, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i})\}$$

Then, applying the basic processor Proc_{Fin}, since the set of pairs is empty, we can trivially conclude the innermost μ-termination of $\mathcal{R}$.

The following example shows that using $\mathsf{iTCAP}^\mu_{\mathcal{R},u}$ provides a better approximation of the ICS-DG than using $\mathrm{TCAP}^\mu_{\mathcal{R}}$ for noncollapsing pairs.

Example 52

Consider the following TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{f}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{f}(x,x,x) \\ \mathtt{c} & \to & \mathtt{a} \\ \mathtt{c} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$. There are two ICS-dependency pairs:

$$\begin{array}{rcl} \mathtt{F}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{F}(x,x,x) \\ \mathtt{F}(\mathtt{a},\mathtt{b},x) & \to & x \end{array}$$

$\mathcal{R}$ is not innermost μ-terminating:

$$\mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \mathtt{F}(\mathtt{a},\underline{\mathtt{c}},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \underline{\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c})} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu,\mathsf{i}),\mu^\sharp} \mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \cdots$$

In order to build the ICS-DG, since there are no hidden terms and therefore $\mathcal{S}_\sharp$ is empty, we only have to check if

$$\mathsf{iTCAP}^\mu_{\mathcal{R},u}(\mathtt{F}(x,x,x)) = \mathsf{iTCAP}^\mu_{\mathcal{R},\mathtt{F(a,b,x)}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x''',x'',x)$$

unifies with $\mathtt{F}(\mathtt{a},\mathtt{b},y)$ so, we get a cycle and the same conclusion would be obtained with $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{F}(x,x,x))$. However, if we use $\mu(\mathtt{f}) = \{1,3\}$, the system now is innermost μ-terminating (the collapsing pair now disappears) but if we use the $\mathrm{TCAP}^\mu_{\mathcal{R}}$ we get $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x''',x'',x)$, which again unifies with $\mathtt{F}(\mathtt{a},\mathtt{b},x)$ and we obtain a spurious cycle. By using $\mathsf{iTCAP}^\mu_{\mathcal{R},u}$, we obtain $\mathsf{iTCAP}^\mu_{\mathcal{R},\mathtt{F(a,b,x)}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x,x,x)$ (since there are no migrating variables now) which does not unify with $\mathtt{F}(\mathtt{a},\mathtt{b},y)$. Now, innermost μ-termination can be easily proved since there are no cycles in the ICS-DG.

After showing that $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}$ provides a better approximation of the ICS-DG for noncollapsing pairs, we are going to show that for the collapsing pairs this is not true since we may get an underestimation of the graph and conclude a false result.

Example 53

Consider the following TRS $\mathcal{R}$ which is a variant of Example 52:

$$\begin{aligned} \mathtt{f}(\mathtt{a},\mathtt{b},x) &\to \mathtt{g}(\mathtt{f}(x,x,x)) \\ \mathtt{g}(x) &\to x \\ \mathtt{c} &\to \mathtt{a} \\ \mathtt{c} &\to \mathtt{b} \end{aligned}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ and $\mu(\mathtt{g}) = \varnothing$. There are two ICS-dependency pairs:

$$\mathtt{F}(\mathtt{a},\mathtt{b},x) \to \mathtt{G}(\mathtt{f}(x,x,x)) \tag{6.4}$$

$$\mathtt{G}(x) \to x \tag{6.5}$$

$\mathcal{R}$ is not innermost μ-terminating:

$$\mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \mathtt{F}(\mathtt{a},\underline{\mathtt{c}},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \underline{\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c})} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu),\mu^\sharp,\mathsf{i}} \underline{\mathtt{G}(\mathtt{f}(\mathtt{c},\mathtt{c},\mathtt{c}))} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu),\mu^\sharp,\mathsf{i}} \mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \cdots$$

We have $\mathcal{S}_\sharp = \{\mathtt{f}(x,x,x) \to \mathtt{F}(x,x,x)\}$. For the pair (6.4) $\in \mathsf{iDP}_\Sigma(\mathcal{R},\mu)$, there is an obvious arc from (6.4) to (6.5). With the only collapsing pair (6.5), since we do not have any information in $\mathcal{S}_\sharp$ about migrating variables, we have to use $\mathrm{TCAP}^{\mu}_{\mathcal{R}}$. In this way, we have that $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x'',x',x)$ unifies with $\mathtt{F}(\mathtt{a},\mathtt{b},y)$ and we obtain an arc from (6.5) to (6.4), thus obtaining the existing cycle {(6.5)-(6.4)}. With $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}$, no variable would be renamed and we would not obtain the arc.

Example 54

Continuing with Example 28, since $\mathsf{iTCAP}^{\mu}_{\mathcal{R},\mathtt{F}(c(x),x)}(\mathtt{F}(x,x)) = \mathtt{F}(x,x)$ and $\mathtt{F}(c(y),y)$ do not unify we conclude that the ICS-dependency graph for the CS-TRS $(\mathcal{R},\mu)$ in Example 28 contains no cycles.

Example 55

Consider the following orthogonal TRS $\mathcal{R}$ which is a variant of an example in [Bor03]:

$$\mathtt{from}(x) \to \mathtt{cons}(x,\mathtt{from}(\mathtt{s}(x)))$$

$$\begin{aligned}
\mathtt{sel}(\mathtt{0}, \mathtt{cons}(x, xs)) &\to x \\
\mathtt{sel}(\mathtt{s}(y), \mathtt{cons}(x, xs)) &\to \mathtt{sel}(y, xs) \\
\mathtt{minus}(x, \mathtt{0}) &\to x \\
\mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) &\to \mathtt{minus}(x, y) \\
\mathtt{quot}(\mathtt{0}, \mathtt{s}(y)) &\to \mathtt{0} \\
\mathtt{quot}(\mathtt{s}(x), \mathtt{s}(y)) &\to \mathtt{s}(\mathtt{quot}(\mathtt{minus}(x, y), \mathtt{s}(y))) \\
\mathtt{zWquot}(\mathtt{nil}, \mathtt{nil}) &\to \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{cons}(x, xs), \mathtt{nil}) &\to \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{nil}, \mathtt{cons}(x, xs)) &\to \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{cons}(x, xs), \mathtt{cons}(y, ys)) &\to \mathtt{cons}(\mathtt{quot}(x, y), \mathtt{zWquot}(xs, ys))
\end{aligned}$$

together with $\mu(\mathtt{cons}) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for all other symbols f. According to [GM02b], innermost μ-termination of $\mathcal{R}$ implies its μ-termination as well. The set $\mathsf{iDP}(\mathcal{R}, \mu)$ is:

$$\begin{aligned}
\mathtt{MINUS}(\mathtt{s}(x), \mathtt{s}(y)) &\to \mathtt{MINUS}(x, y) && (6.6)\\
\mathtt{QUOT}(\mathtt{s}(x), s(y)) &\to \mathtt{MINUS}(x, y) && (6.7)\\
\mathtt{QUOT}(\mathtt{s}(x), s(y)) &\to \mathtt{QUOT}(\mathtt{minus}(x, y), \mathtt{s}(y)) && (6.8)\\
\mathtt{SEL}(\mathtt{s}(y), \mathtt{cons}(x, xs)) &\to \mathtt{SEL}(y, xs) && (6.9)\\
\mathtt{SEL}(\mathtt{s}(y), \mathtt{cons}(x, xs)) &\to xs && (6.10)\\
\mathtt{ZWQUOT}(\mathtt{cons}(x, xs), \mathtt{cons}(y, ys)) &\to \mathtt{QUOT}(x, y) && (6.11)
\end{aligned}$$

The unhiding TRS $\mathsf{unh}(\mathcal{R}, \mu)$ consists of:

$$\begin{aligned}
\mathtt{from}(\mathtt{s}(x)) &\to \mathtt{FROM}(\mathtt{s}(x)) && (6.12)\\
\mathtt{zWquot}(x, y) &\to \mathtt{ZWQUOT}(x, y) && (6.13)\\
\mathtt{zWquot}(x, y) &\to x && (6.14)\\
\mathtt{zWquot}(x, y) &\to y && (6.15)
\end{aligned}$$

We can define the following initial CS problem:

$$\tau_0 = (\mathsf{iDP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp, \mathbf{i})$$

The $\mathsf{EIDG}(\mathcal{R}, \mu) = \mathsf{EIG}(\mathsf{iDP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp)$ of the CS problem τ_0 is shown in Figure 6.2. If now we apply the SCC processor we get the following CS subproblems:

$$\mathsf{Proc}_{SCC}(\tau_0) = \{(\{6.6\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}), (\{6.8\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}), (\{6.9\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i})\}$$

We will continue with these subproblems in Example 61.

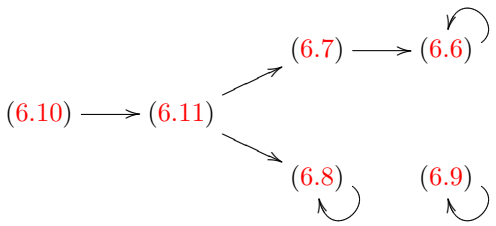


Figure 6.2: Estimated innermost CS-dependency graph for Example 55

6.2 Usable Rules

An interesting feature in the treatment of innermost termination problems using the DP approach is that, since the variables in the right-hand side of the dependency pairs are in normal form, the rules that can be used to connect consecutive dependency pairs are usually a proper subset of the rules in the TRS. This leads to the notion of *usable rules* [AG00, Definition 32], which simplifies the proofs of innermost termination of rewriting. We adapt this notion to the context-sensitive setting.

Definition 56 (Basic Usable CS-Rules [AL07]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For any symbol f let $Rules(\mathcal{R}, f)$ be the set of rules of $\mathcal{R}$ defining f and such that the left-hand side l has no proper μ-replacing $\mathcal{R}$-redex. For any term t, the set of* basic *usable CS-rules $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is as follows:*

$$\begin{array}{lcl} \mathbf{U}_0(\mathcal{R}, \mu, x) & = & \varnothing \\ \mathbf{U}_0(\mathcal{R}, \mu, f(t_1, \ldots, t_n)) & = & Rules(\mathcal{R}, f) \cup \bigcup_{i \in \mu(f)} \mathbf{U}_0(\mathcal{R}', \mu, t_i) \cup \bigcup_{l \to r \in Rules(\mathcal{R}, f)} \mathbf{U}_0(\mathcal{R}', \mu, r) \end{array}$$

where $\mathcal{R}' = \mathcal{R} - Rules(\mathcal{R}, f)$. Consider now another TRS $\mathcal{P}$. Then, $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) = \bigcup_{l \to r \in \mathcal{P}} \mathbf{U}_0(\mathcal{R}, \mu, r)$. Obviously, $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) \subseteq \mathcal{R}$ for all TRSs $\mathcal{P}$ and $\mathcal{R}$.

Interestingly, although our definition is a straightforward extension of the classical one (which just takes into account that μ-rewritings are possible only on μ-replacing subterms), some subtleties arise due to the presence of *non-μ-conservative* rules, i.e., rules with migrating variables.

Basic usable rules $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ in Definition 56 can be used instead of $\mathcal{R}$ when dealing with innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chains associated to μ-conservative TRSs $\mathcal{P}$ provided that $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ is also μ-conservative. This is proved in Theorem 58 below which uses μ-reduction pairs.

A reduction pair $(\gtrsim, \sqsupset)$ consists of a stable and monotonic quasi-order $\gtrsim$, and a stable and well-founded order $\sqsupset$ satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$

[KNT99]. We say that $(\gtrsim, \sqsupset)$ is monotonic if $\sqsupset$ is monotonic. Reduction pairs are used in the DP approach to witness the absence of infinite chains of (dependency) pairs by finding a *reduction pair* $(\gtrsim, \sqsupset)$ that is compatible with the rules and the pairs: $l \gtrsim r$ for all rewrite rules $l \to r$ and $u \sqsupset v$ for all pairs $u \to v$. In the DP framework [GTS04, GTSF06, Thi07] (but also in [GAO02, HM04, HM05, HM07]), they are used to obtain *smaller* sets of pairs $\mathcal{P}' \subset \mathcal{P}$ by removing the *strict* pairs, i.e., those pairs $u \to v \in \mathcal{P}$ such that $u \sqsupset v$ (in this case, all other pairs $u \to v$ that are not strict must be compatible with the quasi-order $\gtrsim$, i.e., $u \gtrsim v$ must hold).

Stability is required both for $\gtrsim$ and $\sqsupset$ because, although we only check the left- and right-hand sides of the rewrite rules $l \to r$ (with $\gtrsim$) and pairs $u \to v$ (with $\gtrsim$ or $\sqsupset$), the chains of pairs involve *instances* $\sigma(l)$, $\sigma(r)$, $\sigma(u)$, and $\sigma(v)$ of rules and pairs, and we aim at concluding that $\sigma(l) \gtrsim \sigma(r)$ and also that $\sigma(u) \gtrsim \sigma(v)$ or $\sigma(u) \sqsupset \sigma(v)$.

Monotonicity is required for $\gtrsim$ to deal with the application of rules $l \to r$ to an arbitrary depth in terms. Since the pairs are 'applied' only at the root level, no monotonicity is required for $\sqsupset$ (but, for this reason, we cannot compare the rules in $\mathcal{R}$ using $\sqsupset$).

In our setting, since we are interested in μ-rewriting steps only, we can relax the *monotonicity* requirements as follows.

Definition 57 (μ-Reduction Pair [AGL06]) *Let Σ be a signature and $\mu \in M_\Sigma$. A* μ-reduction pair $(\gtrsim, \sqsupset)$ *consists of a stable and μ-monotonic quasi-order $\gtrsim$ and a well-founded stable relation $\sqsupset$ on terms in $\mathcal{T}(\Sigma, \mathcal{X})$ which are compatible, i.e., $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$.*

The following theorem formalizes a processor to remove pairs from $\mathcal{P}$ by using μ-reduction pairs and usable CS-rules.

Theorem 58 (Reduction Pair Processor with Usable Rules [AL11]) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $(\gtrsim, \sqsupset)$ be a μ-reduction pair such that*

1. *$\mathcal{P}$ and $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ are μ-conservative,*
2. *$\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$,*

Let $\mathcal{P}_\sqsupset = \{u \to v \in \mathcal{P} \mid u \sqsupset v\}$. Then, the processor Proc_{UR} given by

$$\mathsf{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}), \varnothing, \mu, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound.

Note that the processor is only sound because we refine the result to be applied only to the set of usable rules instead of over the whole set of rules as in standard rewriting [GTSF06] or even for context-sensitive rewriting in [AEF+08, GL10]. In this way, (i.e. by taking all the rules in $\mathcal{R}$), the processor would also be complete, that is:

$$\mathsf{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{R}, \varnothing, \mu, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound and complete. However, since complete processors are useful for disproving termination, we pay more attention to being more precise with the soundness.

Unfortunately, dealing with non-μ-conservative pairs, considering the basic usable CS-rules does *not* ensure a correct approach.

Example 59

Consider again the TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c}(\mathtt{b}) \\ \mathtt{f}(\mathtt{c}(x), x) & \to & \mathtt{f}(x, x) \end{array}$$

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{c}) = \varnothing$. There are *two* non-$\mu$-conservative ICSDPs (note that $\mu^\sharp(\mathtt{F}) = \mu(\mathtt{f}) = \{1\}$):

$$\begin{array}{rcl} \mathtt{F}(\mathtt{c}(x), x) & \to & \mathtt{F}(x, x) \\ \mathtt{F}(\mathtt{c}(x), x) & \to & x \end{array}$$

and only one cycle in the ICS-DG:

$$\{\mathtt{F}(\mathtt{c}(x), x) \to \mathtt{F}(x, x)\}$$

Note that $\mathbf{U}_0(\mathcal{R}, \mu, \mathtt{F}(x, x)) = \varnothing$. Since this ICSDP is strictly compatible with, e.g., an LPO, we would conclude the innermost μ-termination of $\mathcal{R}$. However, this system is *not* innermost μ-terminating:

$$\mathtt{f}(\underline{\mathtt{b}}, \mathtt{b}) \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f}(\mathtt{c}(\mathtt{b}), \mathtt{b})} \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{b}}, \mathtt{b}) \hookrightarrow_{\mathsf{i}} \cdots$$

The problem is that we have to take into account the special status of variables in the right-hand side of a non-μ-conservative ICSDP. Instances of such variables are *not* guaranteed to be μ-normal forms. Furthermore, μ-conservativeness of $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ cannot be dropped either since we could infer an incorrect result as shown by the following example.

Example 60

Consider the TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c}(\mathtt{b}) \\ \mathtt{f}(\mathtt{c}(x), x) & \to & \mathtt{f}(\mathtt{g}(x), x) \\ \mathtt{g}(x) & \to & x \end{array}$$

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{g}) = \mu(\mathtt{c}) = \varnothing$. There is only one μ-conservative cycle:

$$\{\mathtt{F}(\mathtt{c}(x), x) \to \mathtt{F}(\mathtt{g}(x), x)\}$$

having only one usable (but non-μ-conservative!) rule $\mathtt{g}(x) \to x$. This is compatible with the μ-reduction pair induced by the following polynomial interpretation:

$$[\mathtt{f}](x, y) = 0 \qquad [\mathtt{c}](x) = x + 1 \qquad [\mathtt{g}](x) = x \qquad [\mathtt{F}](x, y) = x$$

However the system is not innermost μ-terminating:

$$\underline{\mathtt{f(c(b),b)}} \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{g(b)}},\mathtt{b}) \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{b}},\mathtt{b}) \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f(c(b),b)}} \hookrightarrow_{\mathsf{i}} \cdots$$

Nevertheless, Theorem 58 is useful to improve the proofs of termination of innermost *CSR* as the following example shows.

Example 61

Consider again the TRS $\mathcal{R}$ in Example 55. As we have seen, the initial CS problem can be decomposed in the following three:

$$\tau_1 = (\{6.6\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{6.16}$$
$$\tau_2 = (\{6.8\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{6.17}$$
$$\tau_3 = (\{6.9\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{6.18}$$

Problems τ_1 and τ_3 can be solved by using the subterm processor (see [GL10]). However, without the notion of usable rules, τ_2 is difficult to solve. The pair (6.8) is μ-conservative and the obtained usable rules

$$\{\mathtt{minus}(x, 0) \to x,\ \ \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{minus}(x, y)\}$$

are also μ-conservative. According to Theorem 58, we can apply the usable rules processor $\mathsf{Proc}_{UR}(\tau_2)$ and get the following problem:

$$\tau_4 = (\varnothing, \{\mathtt{minus}(x, 0) \to x, \mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) \to \mathtt{minus}(x, y)\}, \varnothing, \mu^\sharp, \mathbf{i})$$

by using this polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{minus}](x,y) & = & x & & [\texttt{0}] & = & 0 \\ [\texttt{s}](x) & = & x+1 & & [\texttt{QUOT}](x,y) & = & x \end{array}$$

Then, by applying $\mathsf{Proc}_{Fin}(\tau_4)$, since the set of pairs is empty, we can conclude the innermost μ-termination of Example 55. Furthermore, since the system is orthogonal, we have also concluded its μ-termination.

6.3 Usable Arguments for *CSR*

Since in innermost reductions, matching substitutions are always normalized, in an innermost sequence $t_1 \overset{p_1}{\rightarrow}_i t_2 \overset{p_2}{\rightarrow}_i \cdots \overset{p_n}{\rightarrow}_i t_{n+1}$ starting at root position (i.e., $p_1 = \Lambda$), every redex $t_j|_{p_j}$ for $j > 1$ comes from a defined symbol introduced after applying a rule $l_k \rightarrow r_k$ in a previous step $k < j$. Hence the set of arguments which are reduced can be handled by looking for defined symbols in right-hand sides of the involved rules $l \rightarrow r$.

In [Fer05], Fernández defines the notion of *usable arguments* for a function symbol when proving innermost termination. The idea is that, in innermost sequences, some arguments are not relevant for proving termination.

Example 62

Consider the following TRS $\mathcal{R}$:

$$\begin{array}{rcl} \texttt{f}(\texttt{s}(x),\texttt{s}(x)) & \rightarrow & \texttt{f}(x,\texttt{g}(x)) \\ \texttt{g}(\texttt{s}(x)) & \rightarrow & \texttt{g}(x) \end{array}$$

No innermost sequence starting at root position takes into account the first argument of `f` nor the argument of `g` in the rhs. The reason is that, since an innermost redex is an argument normalized redex, that means that all variables (e.g. x) of the applied rule are normalized and cannot be reduced. Only the second argument $\texttt{g}(x)$ of `f` in the right-hand side of the first rule could be innermost reduced after applying it.

Taking these usable arguments into consideration might be helpful in proofs of innermost termination, since they impose weaker monotonicity requirements. For instance, when using polynomial orders, we can use even negative or rational coefficients to interpret the symbols that do not need to be monotonic.

As Fernández noted, the set of usable arguments can be seen as a replacement map that specifies the arguments to be reduced. In her approach, proving the μ-termination of a TRS $\mathcal{R}$ implies the innermost termination of $\mathcal{R}$ if $\mu(f)$=$\mathbf{UA}(f, \mathcal{R}, R)$ for all $f \in \Sigma$ where R only contains rules such that all left-hand sides are argument normalized and $\mathbf{UA}(f, \mathcal{R}, R)$ corresponds to the usable arguments.

Following Fernández's ideas, in the innermost context-sensitive setting (for a given replacement map μ), we could *relax* μ-monotonicity requirements by taking into account that reductions only take place on μ-replacing positions of the right-hand sides of the rules that are rooted by a defined symbol.

We have adapted Fernández's ideas to *CSR* in [AL09]. In sharp contrast to the unrestricted case, we need to take into account that, in innermost *CSR*, a redex does not need to be argument normalized. Only argument μ-normalization can be assumed. Thus, non-μ-replacing subterms may contain redexes that can be reduced later on if they come to a μ-replacing position. The following result is an obvious fact inherent in the innermost strategy for *CSR*.

Proposition 63 *[AL09] A CS-TRS $(\mathcal{R}, \mu)$ is innermost μ-terminating iff $\mathcal{R}'$ is innermost μ-terminating, where $\mathcal{R}' \subseteq \mathcal{R}$ contains all rules $l \to r \in \mathcal{R}$ such that l is argument μ-normalized.*

In the following, we assume that all CS-TRS $(\mathcal{R}, \mu)$ are argument μ-normalized, i.e., for all rules $l \to r$ in $\mathcal{R}$, l is argument μ-normalized. Proposition 63 ensures that this entails no lack of generality regarding our research on innermost termination of *CSR*.

The straightforward adaptation of Fernandez's criterion to *CSR* yields the following definition: the usable CS-arguments for a function symbol $f \in \Sigma$ are those arguments with a μ-replacing subterm rooted by a defined symbol in some right-hand side of a pair or usable rule.

Definition 64 (Basic Usable CS-Arguments [AL09]) *Let $(\mathcal{R}, \mu) = ((\mathcal{C} \uplus \mathcal{D}, R), \mu)$ be a CS-TRS and $\mathcal{P}$ be a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument μ-normalized. The* basic usable CS-arguments *for a function symbol $f \in \Sigma$ are defined as $\mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P}) = \{i \in \mu(f) \mid \exists u \to v \in \mathcal{P} \cup \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}),\ \exists p, p' \in \mathcal{P}os^\mu(v)$ s.t. $root(v|_{p'}) = f$, $root(v|_p) \in \mathcal{D}$, $p'.i \leqslant p$, $u \not\trianglerighteq_\mu v|_p\}$.*

Note that the replacement map given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for all $f \in \Sigma$ is more restrictive than μ: $\mu'(f) \subseteq \mu(f)$ for all $f \in \Sigma$.

Innermost μ-termination can be proved as innermost μ'-termination for μ' given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for all $f \in \Sigma$ whenever $\mathcal{R}$ is μ-conservative.

Theorem 65 *[AL09] Let $\mathcal{R}$ be a μ-conservative TRS for $\mu \in M_\mathcal{R}$. Let μ' be given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, R)$ for every $f \in \Sigma$. If $\mathcal{R}$ is innermost μ'-terminating, then $\mathcal{R}$ is innermost μ-terminating.*

The following example shows the need of restricting the attention to μ-conservative TRSs.

Example 66

Consider the CS-TRS $\mathcal{R}$:

$$\begin{array}{rcl} \mathtt{f}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{f}(x,x,x) \\ \mathtt{c} & \to & \mathtt{a} \\ \mathtt{c} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1, 2\}$. If we try to apply Theorem 65 to prove innermost μ-termination of $\mathcal{R}$, we obtain $\mu'(\mathtt{f}) = \varnothing$ and the ICS-dependency graph has no cycle thus concluding the innermost μ-termination of $\mathcal{R}$. However, $\mathcal{R}$ is *not* innermost μ-terminating:

$$\underline{\mathtt{f(a,b,c)}} \hookrightarrow_\mathsf{i} \mathtt{f}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_\mathsf{i} \mathtt{f}(\mathtt{a},\underline{\mathtt{c}},\mathtt{c}) \hookrightarrow_\mathsf{i} \underline{\mathtt{f(a,b,c)}} \hookrightarrow_\mathsf{i} \cdots$$

Note that the first rule of $\mathcal{R}$ is not μ-conservative.

We can adapt the use of usable CS-arguments to be applied in proofs of innermost μ-termination in our CSDP framework. We do that by providing a new processor.

Theorem 67 (RP Processor with Usable Rules and Arguments [AL11]) *Let $\mathcal{R} = (\Sigma, R)$, $\mathcal{P} = (\mathcal{G}, P)$, and $\mathcal{S} = (\mathcal{H}, S)$ be TRSs, $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$, and $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $\mu_\mathcal{A}(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for all $f \in \Sigma \cup \mathcal{G}$ and $(\gtrsim, \sqsupset)$ be a $\mu_\mathcal{A}$-reduction pair such that*

1. *$\mathcal{P}$ and $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ are μ-conservative,*
2. *$\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$,*

Let $\mathcal{P}_\sqsupset = \{u \to v \in \mathcal{P} \mid u \sqsupset v\}$. Then, the processor Proc_{Fer} given by

$$\mathsf{Proc}_{Fer}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}), \varnothing, \mu_\mathcal{A}, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound.

Note that, as in Theorem 58, the use of usable CS-rules avoids completeness of the processor.

Now, for a given CS-TRS $(\mathcal{R}, \mu)$ that satisfies the conditions of Theorem 67, we can prove its innermost μ-termination by relaxing μ-monotonicity requirements.

6.4 Narrowing Transformation

Although the use of function TCap leads to a good approximation of the graph, it can lead to *overestimate* the arcs that connect two dependency pairs. As already observed by Arts and Giesl for the standard innermost case [AG00], in our setting the overestimation comes when a (noncollapsing) pair $u_i \to v_i$ is followed in a chain by a second one $u_{i+1} \to v_{i+1}$ and v_i and u_{i+1} are not directly unifiable, i.e., at least one (innermost) μ-rewriting step is needed to (innermost) μ-reduce $\sigma(v_i)$ to $\sigma(u_{i+1})$. Then, the (innermost) μ-reduction from $\sigma(v_i)$ to $\sigma(u_{i+1})$ requires at least one step, i.e., we always have $\sigma(v_i) \hookrightarrow_{\mathcal{R},\mu^\sharp} \sigma(v_i') \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$. Furthermore, we could discover that v_i has *no μ-narrowings*. In this case, we know that no (innermost) μ-chain starts from $\sigma(v_i)$. A restriction that has to be taken into account when μ-narrowing a noncollapsing pair $u \to v$ is that the μ-replacing variables in v have to be μ-replacing in u as well (this corresponds with the notion of μ-conservativeness), but furthermore, they cannot be both μ-replacing and non-μ-replacing at the same time. This corresponds to the following definition.

Definition 68 (Strongly μ-conservative [GLU08]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. A rule $l \to r$ is strongly μ-conservative if it is μ-conservative and $\mathcal{V}ar^\mu(l) \cap \mathcal{V}ar^{\not\mu}(l) = \mathcal{V}ar^\mu(r) \cap \mathcal{V}ar^{\not\mu}(r) = \varnothing$.*

In [AGL10], we define the following μ-narrowing processor.

Theorem 69 (μ-Narrowing Processor) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})$ be a CS problem. Let $u \to v \in \mathcal{P}$ be such that*

1. *v is linear, and*
2. *for all $u' \to v' \in \mathcal{P}$ (with possibly renamed variables), v and u' do not unify.*

Let $\mathcal{Q} = (\mathcal{P} - \{u \to v\}) \cup \{u' \to v' \mid u' \to v' \text{ is a } \mu\text{-narrowing of } u \to v \text{ w.r.t. } \mathcal{R}\}$. Then, the processor Proc_{narr} given by

$$\mathsf{Proc}_{narr}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t}) = \begin{cases} \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{otherwise} \end{cases}$$

is

1. *sound whenever $u \to v$ is strongly μ-conservative, and*
2. *complete in all cases.*

Of course, μ-narrowing can also be used in proofs of innermost termination of *CSR*. In the standard setting, when using narrowing for proving innermost termination, we do not require the right-hand side of the dependency pair to be narrowed to be linear since the involved substitution σ is normalized. However, in the context-sensitive setting, if the pair to be μ-narrowed is *not strongly μ-conservative*, we cannot ensure that the variables on the right-hand side are μ-normalized, so *we also have to demand linearity.* When dealing with innermost narrowing in context-sensitive rewriting, we can drop the linearity condition if the pair to be μ-narrowed is strongly μ-conservative since all μ-replacing variables in the right-hand side of a pair are instantiated to μ-normal form and μ-reductions cannot take place on them.

Theorem 70 (Innermost μ-Narrowing Processor [AL11]) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $u \to v \in \mathcal{P}$ be such that*

1. *for all $u' \to v' \in \mathcal{P}$ (with possibly renamed variables), v and u' do not unify or they unify by some mgu θ such that one of the terms $\theta(u)$ or $\theta(u')$ is not a μ-normal form.*

Let $\mathcal{Q} = (\mathcal{P} - \{u \to v\}) \cup \{u' \to v' \mid u' \to v' \text{ is a } \mu\text{-narrowing of } u \to v \text{ w.r.t. } \mathcal{R}\}$. Then, the processor Proc_{Inarr} given by

$$\mathsf{Proc}_{Inarr}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{if (1) holds} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is

1. *sound whenever $u \to v$ is strongly μ-conservative, and*
2. *complete in all cases.*

Example 71

Consider the following example:

$$\begin{array}{lcl} \mathtt{f}(\mathtt{s}(x)) & \to & \mathtt{f}(\mathtt{p}(\mathtt{s}(x))) \\ \mathtt{p}(\mathtt{s}(x)) & \to & x \end{array}$$

together with $\mu(\mathtt{f}) = \mu(\mathtt{s}) = \{1\}$ and $\mu(\mathtt{p}) = \varnothing$.

The only ICSDP that could generate a cycle is $\mathtt{F}(\mathtt{s}(x)) \to \mathtt{F}(\mathtt{p}(\mathtt{s}(x)))$. However since the right-hand side $\mathtt{F}(\mathtt{p}(\mathtt{s}(x)))$ does not unify with any (renamed) left-hand side (including itself) and the pair is strongly μ-conservative, we can apply innermost μ-narrowing. Therefore, the pair can be μ-narrowed at position 1 (notice that $\mu(f) = \mu(\mathtt{F}) = \{1\}$) by using the rule $\mathtt{p}(\mathtt{s}(x)) \to x$. Then, the pair is transformed into the pair $\mathtt{F}(\mathtt{s}(y)) \to \mathtt{F}(y)$ that can be easily disregarded by using the subterm criterion[1].

[1] Instead of using in the proof a polynomial interpretation with rationals, like MU-TERM or matrix interpretations like AProVE.

7

Experiments on ICS Rewriting

We have implemented the techniques described in the previous chapters as part of the tool MU-TERM [AGLN10, AGIL07, Luc04], which is a tool that can be used to verify a number of termination properties of (variants of) TRSs. In order to evaluate the techniques that are reported in this part, we have made some benchmarks. We have considered the examples in the Termination Problem Data Base[1] (TPDB) .

7.1 Direct Techniques vs. Transformations

Although there is no special TPDB category for innermost termination of *CSR* (yet), we have considered the examples that are collected in the *CSR* category in order to test our techniques for proving termination of innermost *CSR*. The TPDB v7.0.2 contains 109 examples of CS-TRSs. In order to compare our direct techniques with the transformational approach (see [GM02b, GM02a] for a survey on this topic) where termination of innermost *CSR* for a CS-TRS $(\mathcal{R}, \mu)$ is proved by proving innermost termination of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$, where Θ specifies a particular transformation, we have transformed the set of examples by using Giesl and Middeldorp's correct transformations for proving termination of innermost *CSR* (see [GM02a]) although we use the 'authors-based' notation introduced in [Luc06]:

- GM and C for transformations 1 and 2 for proving termination of *CSR* introduced in [GM04], and
- iGM for the specific transformation for proving termination of innermost *CSR* introduced in [GM02a].

[1] http://www.termination-portal.org/wiki/TPDB

	ICSDPs	Transformations
YES score	95/109	60/109
YES average time	0.7 sec.	1.5 sec.

Table 7.1: Comparison in proofs of termination of innermost *CSR*

	C	GM	iGM
YES score	33	57	42

Table 7.2: Comparing transformations for proving termination of innermost *CSR*

Then, we have proved innermost termination of the set of examples with AProVE [GST06], which is able to prove innermost termination of standard rewriting[2]. The results are summarized in Tables 7.1 and 7.2. Further details can be found here:

http://www.dsic.upv.es/~balarcon/iCSR/benchmarks.html

These are the first known benchmarks that compare transformational techniques vs. direct (DP-based) techniques as well as the existing correct transformations for proving innermost termination of *CSR* among them. From the results in Table 7.1, we can conclude that the use of ICSDPs dramatically improves the performance of previous transformational approaches. Moreover, the results in Table 7.2 show that, quite surprisingly, the iGM transformation (which is in principle the more suitable one for proving innermost termination of *CSR*, according to the theoretical results in [GM02a]) obtains worse results than GM (on average).

Previously, in [AL07], we obtained 70 out of 90 successful proofs versus 44 for transformations (at this time the TPDB v3.2 was used). Without a doubt, the use of ICSDPs were imposed to prove innermost termination of *CSR*. With the recent developments of MU-TERM embracing the DP framework, MU-TERM would solve 77 out of the 90 examples of the previous version of the TPDB. Furthermore, all the examples that can be solved by using transformations, before and now, can be solved by ICSDPs.

[2]Nowadays, AProVE is the best tool for proving termination of innermost rewriting according to the benchmarks obtained in the last termination competition, see http://termcomp.uibk.ac.at

Therefore, from the results in Table 7.1, it is clear that the ICSDP framework is the right choice for proving innermost termination of *CSR* instead of using transformations. The complete set of benchmarks can be also consulted in Appendix A (Chapter 19).

7.2 Relaxing Monotonicity Requirements

We have used the set of examples mentioned in Section 7.1 for our experiments on proving termination of innermost *CSR* by means of a new replacement map that imposes less monotonicity requirements.

We have implemented the use of Theorem 67 to deal with non-μ-conservative systems (but μ-conservative cycles). MU-TERM tries to solve each μ-conservative cycle (with associated μ-conservative usable rules) by using usable CS-arguments as the new replacement map. This implementation of MU-TERM succeeds in the same 95 examples, the same number of examples that we had already solved using ICSDPs. The average time rate does not exhibit substantial differences. Further details can be found here:

http://www.dsic.upv.es/~balarcon/iCSR_UA/benchmarks.html

Although no improvement over the practical use of ICSDPs explained in Section 7.1 is shown, we expect that, in the near future, when we implement nonmonotonic orders in our termination tool MU-TERM, we will be able to take advantage of this technique.

Moreover, we have implemented the use of Corollary 11 in [Fer05] to prove innermost termination of TRSs by proving μ-termination of the CS-TRS obtained after using the usable arguments as replacement map (this was one of the main results in Fernández's paper). The relevance of this result in practice had not yet been tested as no implementation of Fernández's results was available (to our knowledge). In order to evaluate it, we have considered the examples from the TPDB used in the *innermost category*. There are 358 examples. Using usable arguments (we call this tool MU-TERM UA), MU-TERM succeeds in 158 examples. However, we have also implemented the use of (standard) dependency pairs for proving innermost termination (according to [AG00, Theorem 37]) together with the narrowing refinement (we call this tool MU-TERM iDPs) and we are able to prove 199 examples, including all examples solved with Fernández's criterion.

Therefore, it seems that using her result to prove innermost termination of rewriting is not a good idea (at least with the considered set of examples) since we lose some examples and the average time is worse. The results are summarized in Table 7.3. Further details can be found in:

	MU-TERM UA	MU-TERM iDPs
YES score	158	199
YES average time	4.87 sec.	3.31 sec.

Table 7.3: Benchmarks for innermost termination of rewriting

http://www.dsic.upv.es/~balarcon/UA/benchmarks.html

All this shows that we do not obtain any real improvement over the basic technique of dependency pairs for proving innermost termination at least for the set of considered examples.

7.3 Transforming CS-dependency Pairs

We have also implemented (innermost) μ-narrowing in MU-TERM. Due to the possibility of performing an unbounded number of narrowing steps, the μ-narrowing transformation could be infinite (this also occurs in the standard approach). In order to implement the transformation, we have chosen to use *one-step μ-narrowing* only if the obtained innermost context-sensitive dependency graph has less cycles and arcs than the original one. Practically, although several examples can be solved by using narrowing, they can also be solved by using other techniques that MU-TERM implements. Therefore, it is one of the last techniques that MU-TERM tries when proving (innermost) μ-termination. The greatest advantage of using μ-narrowing lies in the possibility of dismissing some CSDPs if they have no μ-narrowings, thus simplifying the proof of termination.

7.4 Termination Competition

Thanks to the developments reported in this thesis and in [AGL10, GL10], MU-TERM 5.07 has proven to be the most powerful tool for proving termination of *CSR* in the *context-sensitive* subcategory of the 2007, 2009, and 2010 editions of the International Competition of Termination Tools[3]. The

[3]See http://www.lri.fr/~marche/termination-competition/2007/, where only AProVE and MU-TERM participated, and http://termcomp.uibk.ac.at/termcomp/ where there were three more tools in the competition: AProVE, Jambox, and VMTL [SG09]. AProVE and MU-TERM solved the same number of examples, but MU-TERM was much faster. The same situation occurred in 2010 (but without Jambox's participation). See Table 7.4 for details.

summarized results are in Table 7.4.

	MU-TERM	AProVE	Jambox	VMTL
	YES (av. time)	YES (av. time)	YES (av. time)	YES (av. time)
2007	68/90 (2.87s)	64/90 (6.90s)	-	-
2009	34/37 (1.27s)	34/37 (3.84s)	28/37 (2.29s)	29/37 (6.71s)
2010	33/37 (0.37s)	33/37 (1.64s)	-	28/37 (4.78s)

Table 7.4: International Termination Competition results on *CSR*

As we showed in Section 5.1, under certain conditions, termination of *CSR* and termination of innermost *CSR* coincide. For this reason, one of the most important aspects of innermost *CSR* is its use for proving termination of *CSR* as part of the CSDP framework. We switch from termination of *CSR* to termination of innermost *CSR* whenever termination is equivalent, for which we can apply the existing processors more successfully. Actually, we proceed in this way in around 50% of the *CSR* termination problems which are proved by MU-TERM 5.0. More precisely, out of the 95 examples that can be proved μ-terminating by MU-TERM, 43 of them are orthogonal and, thus, its μ-termination is proved by proving innermost μ-termination instead.

8

Related Work and Contributions

8.1 Related work

The first work that tried to prove innermost termination of context-sensitive rewriting [Luc01a] analysed the existing transformations for proving termination of context-sensitive rewriting [Luc96, Zan97, GM99] applied to the innermost setting. Only the two transformations of [GM99] were shown correct for that. Later, the first specific transformation for dealing with innermost termination of *CSR* [GM02a] was developed. As we have stated, [AL07] pioneered the development of direct methods for proving termination of innermost context-sensitive rewriting. We extended the context-sensitive dependency pair method of [AGL06] to the innermost *CSR* setting. Since then, many improvements have been introduced. For instance, the definition of innermost CSDP has been improved by introducing the μ-narrowability condition of [LM08].

Example 72

Consider the following CS-TRS $\mathcal{R}$ in [GM02b]:

$$\begin{array}{rcl} \mathtt{f(g(b))} & \to & \mathtt{f(g(a))} \\ \mathtt{f(a)} & \to & \mathtt{f(a)} \\ \mathtt{a} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{g}) = \varnothing$. Then the set of dependency pairs in [AL07] for proving innermost μ-termination of $\mathcal{R}$ was:

$$\mathtt{F(g(b))} \to \mathtt{F(g(a))}$$

and $\mu^\sharp(\mathtt{F}) = \{1\}$. Now, with the new definition of $\mathsf{iDP}(\mathcal{R}, \mu)$, we do not obtain any pair since it does not have the μ-narrowability condition. So, now,

$\mathsf{iDP}(\mathcal{R},\mu) = \varnothing$.

Moreover, thanks to the developments in [GL10] the definition of innermost μ-chain has been improved as well.

Definition 73 (Innermost μ-Chain [AL07]) *Given a CS-TRS $(\mathcal{P},\mu^\sharp)$ of CSDPs associated to a CS-TRS $(\mathcal{R},\mu)$, an innermost $(\mathcal{R},\mathcal{P},\mu^\sharp)$-chain is a sequence of pairs $u_j \to v_j \in \mathcal{P}$ such that there is a substitution σ such that $\sigma(u_j) \in \mathsf{NF}_\mu(\mathcal{R})$ and for all $j \geq 1$,*

1. *$\sigma(v_j) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma(u_{j+1})$, if $u_j \to v_j \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$, and*
2. *if $u_j \to v_j = u_j \to x_j \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$, then there is some $s_j \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $\sigma(x_j) \trianglerighteq_\mu s_j$ and $s_j^\sharp \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma(u_{j+1})$.*

As usual we assume that different occurrences of dependency pairs do not share any variables (if necessary renamings are used). An innermost $(\mathcal{R},\mathcal{P},\mu^\sharp)$-chain is minimal *if for all $u_j \to v_j \in \mathcal{P}$ and $j \geq 1$, $\sigma(v_j)$ is innermost μ-terminating (whenever $u_j \to v_j \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$) and $s_j^\sharp$ is innermost μ-terminating (whenever $u_j \to v_j \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$).*

- In [AL07], the subterm condition and the marking are part of the notion of chain and parametrized by $\mathcal{R}$.
- Now, following [GL10], the subterm condition and the marking are encoded by the unhiding TRS and are explicitly isolated by means of the rules in $\mathcal{S}$ (see Definition 26).

Note that if rules $f(x_1,\ldots,x_k) \to x_i$ for all $f \in \mathcal{D}$ and $i \in \mu(f)$ (where $x_1,\ldots,x_k$ are variables) are used in Definition 24-(1), and rules $f(x_1,\ldots,x_k) \to f^\sharp(x_1,\ldots,x_k)$ for all $f \in \mathcal{D}$ are used in Definition 24-(2), then we have the original notion of innermost chain in [AL07]. Thus, the new definition covers the previous one.

Another important difference is that [AL07] was the adaptation of the DP approach [AG00] to innermost *CSR*: now we have a DP framework in which proof techniques are handled and incorporated to the whole proof as CS processors in an incremental way, improving the performance of [AL07] in practice as we have seen in the previous chapter.

With respect to the graph, several estimations of the dependency graph were investigated in [AG00, HM05, GTS05, Mid01, Mid02]. The first one, which was introduced in [AG00] and which used functions Cap and Ren, was

adapted to *CSR* in [AGL06]. Following this approximation, [AL07] adapted it to the innermost context-sensitive setting. In [AGL10, Subsection 8.2], we adapted Giesl et al.'s TCAP to *CSR* to obtain our latest approximation of the CAP function, $\text{TCAP}^{\mu}_{\mathcal{R}}$. We also use it for innermost *CSR* in the case of connecting collapsing pairs and we have also defined an innermost version $\text{iTCAP}^{\mu}_{\mathcal{R}}$ to be used with noncollapsing pairs. Moreover, we can take advantage of the developments we have made in *CSR*: The estimated CS graph of pairs has evolved over time and has been improved thanks to the notions of hidden symbols, hidden terms, and hiding context and therefore, also the ICS graph.

8.2 Contributions

The results of this part of the thesis are revised and extended versions of the results published in [AL07, AL09], taking into account all the improvements made in the CSDP framework in [AGL10, GL10].

Theoretical Contributions. We have investigated the structure of infinite innermost context-sensitive rewrite sequences starting from (strongly) minimal innermost non-μ-terminating terms (Theorem 21). This knowledge has been used to provide an appropriate definition of innermost context-sensitive dependency pair (Definition 22), and the related notion of innermost chain (Definition 26). We have proved that it can be used to characterize innermost μ-termination (Theorem 27). We have provided a suitable adaptation of Giesl et al.'s *dependency pair framework* to innermost *CSR* by defining appropriate notions of *CS problem* (Definition 30) and *CS processor* (Definition 31). We have described the connection between innermost *CSR* and *CSR* and we have developed a CS processor (Theorem 39) that allows us to switch from one framework to another under some conditions, increasing the power of both frameworks. We have described a number of sound and (most of them) complete CS processors, which can be used in any practical implementation of the ICSDP framework. In particular, we have introduced the notion of (estimated) *innermost context-sensitive (dependency) graph* (Definitions 42 and 47) by using functions to approximate it (Definition 46) and the associated CS processor showing how to automatically prove innermost μ-termination by means of the ICS dependency graph (Theorem 49). We have formulated the notion of basic usable rules showing how to use them in proofs of innermost termination of *CSR* (Definition 56, Theorem 58). We have also shown how to relax monotonicity requirements for proving innermost termination of

context-sensitive rewriting. We have adapted Fernández's approach [Fer05] to be used for proving innermost termination of context-sensitive rewriting as a CS processor (Theorem 67). Narrowing has also been investigated in proofs of innermost *CSR*. It can also be helpful to simplify or restructure the ICS dependency graph and eventually simplify the proof (Theorem 70).

Applications and Practical Impact. We have implemented these ideas as part of the termination tool MU-TERM [AGLN10, AGIL07, Luc04]. The implementation and practical use of the developed techniques yield a novel and powerful framework that improves the current state-of-the-art of methods for proving innermost termination of *CSR*. Actually, ICSDPs were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007, 2009, and 2010 competitions of termination tools.

Up to our contributions, no direct method has been proposed to prove termination of innermost *CSR*. We have extended the DP framework to prove innermost termination of *CSR*. Our benchmarks show that the use of ICSDPs dramatically improves the performance of existing (transformational) methods for proving termination of innermost *CSR*.

As remarked in the introduction, our goal is to apply all these developments in order to deal with termination of Maude programs. Since its computational mechanism can be thought of as a kind of "context-sensitive call by value", we believe that our research is an essential contribution to the development of tools for proving termination of Maude programs.

Part II

Termination of $A \vee C$-Rewriting

9

Infinite $A \vee C$-Rewrite Sequences

Several works have tried to adapt the DP approach [AG00] to rewriting modulo *associative and commutative* (AC) theories [KNT06, KT01, Kus00, MU98, MU04]. The corresponding proof methods, though, cannot be applied to commonly occurring combinations of axioms that fall outside their scope. Regarding E-termination analysis (for a given set of equational axioms E) using *dependency pairs*, Kusakari and Toyama observed that there is no simple extension of DPs to directly deal with $\rightarrow_{R/E}$-computations [KT01, Kus00]. In contrast, several approaches have been developed for $\rightarrow_{R,E}$-computations [GK01, KT01, MU98]. Since $\rightarrow_{R,E} \subseteq \rightarrow_{R/E}$ (but the opposite inclusion does not hold, in general), E-termination cannot be concluded from (R, E)-termination. Actually, Marché and Urbain showed that there are (R, E)-terminating rewrite theories $\mathcal{R}$ which are *not* E-terminating.

Example 74

Consider the following rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where '+' is an AC symbol [MU98]:

$$a + b \rightarrow a + (b + c).$$

Note that $t = a + (b + c)$ is an $\rightarrow_{R,E}$-normal form (hence (R, E)-terminating). However, $t \sim_{AC} (a + b) + c$ which is E-nonterminating.

Giesl and Kapur [GK01] proved the equivalence of both notions of termination with respect to a notion of *extension completion* $\mathcal{E}xt_E(R)$ (see below) of a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ for E *regular* (i.e., $\mathcal{V}ar(u) = \mathcal{V}ar(v)$ for all $u = v$ in E), and *linear* (neither u nor v have repeated variables). For E being a set containing associative or commutative axioms, this notion of extension goes back to Peterson and Stickel [PS81].

Theorem 75 [GK01, Theorem 11] *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory with E a regular and linear equational theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, t starts an infinite $\rightarrow_{R/E}$-reduction if and only if t starts an infinite $\rightarrow_{\mathcal{E}xt_E(R),E}$-reduction. Therefore, $\mathcal{R}$ is E-terminating if and only if $\rightarrow_{\mathcal{E}xt_E(R),E}$ is terminating.*

9.1 Combination of Associative and Commutative Theories

Let E be a set of equations that has the modular decomposition $E = \bigcup_{f \in \Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$, where:

- A_f is the associativity axiom $f(f(x, y), z) = f(x, f(y, z))$,
- C_f is the commutativity axiom $f(x, y) = f(y, x)$.

We also define $\Sigma = \Sigma_A \uplus \Sigma_C \uplus \Sigma_{AC} \uplus \Sigma_\varnothing$ where $f \in \Sigma_A \Leftrightarrow E_f = \{A_f\}$, $f \in \Sigma_C \Leftrightarrow E_f = \{C_f\}$, $f \in \Sigma_{AC} \Leftrightarrow E_f = \{A_f, C_f\}$, $f \in \Sigma_\varnothing \Leftrightarrow E_f = \varnothing$. In the following, we often say that a symbol $f \in \Sigma$ *is associative* iff $f \in \Sigma_A \cup \Sigma_{AC}$.

Definition 76 ($A \vee C$-Rewrite Theory [ALM10]) *An equational theory $E = \bigcup_{f \in \Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$ is called an* $A \vee C$-theory. *A rewrite theory $\mathcal{R} = (\Sigma, E, R)$ such that E is an $A \vee C$-theory, is called an* $A \vee C$-rewrite theory.

To deal with rewriting modulo $A \vee C$-theories by using (R, E)-rewriting we have to extend R by following [PS81, Definition 10.4]:

$$\begin{array}{rcl} \mathcal{E}xt_{AC}(R) & = & R \cup \{f(l, w) \rightarrow f(r, w) \mid l \rightarrow r \in R, f = root(l) \in \Sigma_{AC}\} \\ \mathcal{E}xt_A(R) & = & R \cup \{f(l, w) \rightarrow f(r, w), f(w, l) \rightarrow f(w, r), f(z, f(l, w)) \rightarrow f(z, f(r, w)) \\ & & \mid l \rightarrow r \in R, f = root(l) \in \Sigma_A\} \\ \mathcal{E}xt_C(R) & = & R \end{array}$$

where w and z are fresh variables which do not occur in the original rule of R. Therefore, given an $A \vee C$-theory E, we let:

$$\mathcal{E}xt_E(R) = \mathcal{E}xt_{AC}(R) \cup \mathcal{E}xt_A(R) \cup \mathcal{E}xt_C(R).$$

Note that $R \subseteq \mathcal{E}xt_E(R)$.

Example 77

Consider the following TRS $\mathcal{R}$:

$$f(x, x) \rightarrow f(0, 0)$$

where $f \in \Sigma_{AC}$. Hence, $\mathcal{E}xt_{AC}(R)$ only adds the following rule to $\mathcal{R}$:

$$f(f(x,x),y) \quad \to \quad f(f(0,0),y)$$

Proposition 78 (E-Termination Preserved under E-Equivalence [ALM10]) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory and $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If $t \sim_E s$, then t is E-terminating if and only if s is E-terminating.*

Proposition 78 does *not* hold if we change E-termination by (R, E)-termination (see Example 74). However, as a consequence of Theorem 75 and Proposition 78, we have:

Corollary 79 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory such that E is a set of regular and linear equations and $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If $t \sim_E s$, then t is $(\mathcal{E}xt_E(R), E)$-terminating if and only if s is $(\mathcal{E}xt_E(R), E)$-terminating.*

As a corollary of Theorem 75, we have the following.

Corollary 80 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, t is E-terminating if and only if it is $(\mathcal{E}xt_E(R), E)$-terminating.*

In the following, we begin the analysis of infinite E-rewrite sequences according to the schema of [HM04] as we have made in Part I for infinite innermost context-sensitive sequences. We aim at providing an appropriate notion of minimal E-nonterminating term (for $A \vee C$-theories E) which allows us to reach a result similar to Proposition 11.

9.2 Minimal E-nonterminating Terms

The following notion of minimal E-nonterminating term is implicit in [GK01, proof of Theorem 16]. Similar definitions can be found in [KNT06, KT01, Kus00, MU04].

Definition 81 (Minimal E-Nonterminating Term [GK01]) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. An E-nonterminating term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is said to be* minimal *(written $t \in \mathcal{T}_{\infty,R,E}$) if every strict subterm s of t (i.e., $t \rhd s$) is $(\mathcal{E}xt_E(R), E)$-terminating.*

Remark 82 In Definition 81, if we assume that E is linear and regular (like $A\vee C$-theories), then, by Theorem 75, we could equivalently start by saying that t is $(\mathcal{E}xt_E(R), E)$-nonterminating. This leads to a more symmetric definition, which we often use in the following without further comment. ■

Every E-nonterminating term s contains a minimal E-nonterminating subterm $t \in \mathcal{T}_{\infty,R,E}$ (this is stated without proof in [GK01, proof of Theorem 16]). Moreover, Giesl and Kapur's minimality of terms is preserved under *inner* $\to_{\mathcal{E}xt_E(R),E}$-reductions.

Note that, if E is an $A\vee C$-equational theory, then $root(t) \in \mathcal{D}$ whenever $t \in \mathcal{T}_{\infty,R,E}$. As remarked by Giesl and Kapur (see also Example 90 below) this is not true for arbitrary equational theories. The problem with Giesl and Kapur's Definition 81 is that minimality is *not* preserved under E-equivalence.

Example 83

Consider again the TRS $\mathcal{R}$ in Example 77.
Following [GK01], the term $f(f(1,0),0) \in \mathcal{T}_{\infty,R,E}$ since it is AC-nonterminating

$$f(f(1,0),0) \sim_{AC} f(1,f(0,0)) \sim_{AC} f(f(0,0),1) \xrightarrow{\Lambda}_R f(f(0,0),1)\cdots$$

but its strict subterms $f(1,0)$, 1 and 0 are AC-terminating. However, the root step with $\sigma(l) = \sigma(f(f(x,x),y) = f(f(0,0),1)$ shows that $\sigma(l) \notin \mathcal{T}_{\infty,R,E}$ since $f(0,0)$ is AC-nonterminating.

Example 84

Consider the following TRS $\mathcal{R}$:

$$f(x,x) \quad \to \quad f(0,f(1,2)) \tag{9.1}$$

where $f \in \Sigma_{AC}$. Hence, $\mathcal{E}xt_{AC}(R)$ only adds the following rule to $\mathcal{R}$:

$$f(f(x,x),y) \quad \to \quad f(f(0,f(1,2)),y) \tag{9.2}$$

Note that $t = f(f(0,1),f(0,f(1,2)))$ is $(\mathcal{E}xt_{AC}(R), AC)$-nonterminating:

$$\begin{array}{rl}
\underline{f(f(0,1),f(0,f(1,2)))} & \\
\sim_A & f(0,\underline{f(1,f(0,f(1,2)))}) \\
\sim_A & f(0,f(\underline{f(1,0)},f(1,2))) \\
\sim_C & f(0,\underline{f(f(0,1),f(1,2))}) \\
\sim_A & \underline{f(0,f(0,f(1,f(1,2))))} \\
\sim_A & \underline{f(f(0,0),f(1,f(1,2)))} \\
\xrightarrow{\Lambda}_{\mathcal{E}xt_{AC}(R)} & \underline{f(f(0,f(1,2)),f(1,f(1,2)))} \\
\to_{\mathcal{E}xt_{AC}(R),AC} & \cdots
\end{array}$$

Since $f(0,1)$ and $f(0,f(1,2))$ are in $(\mathcal{E}xt_{AC}(R),AC)$-normal form, we have that $t \in \mathcal{T}_{\infty,R,AC}$. However, $t' = f(f(0,0),f(1,f(1,2)))$, which is AC-equivalent to t (i.e., $t \sim_{AC} t'$), is AC-nonterminating, but it is *not* minimal because its strict subterm $f(1,f(1,2)))$ is $(\mathcal{E}xt_{AC}(R),AC)$-nonterminating:

$$
\begin{array}{lll}
\underline{f(1,f(1,2))} & \sim_A & \underline{f(f(1,1),2)} \\
& \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_{AC}(R)} & \underline{f(f(0,f(1,2)),2)} \\
& \sim_A & f(0,\underline{f(f(1,2),2)}) \\
& \sim_A & \underline{f(0,f(1,f(2,2)))} \\
& \sim_A & \underline{f(f(0,1),f(2,2))} \\
& \sim_C & \underline{f(f(2,2),f(0,1))} \\
& \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_{AC}(R)} & f(\underline{f(0,f(1,2))},f(0,1)) \\
& \sim_A & \underline{f(f(f(0,1),2)),f(0,1))} \\
& \sim_C & \underline{f(f(0,1),f(f(0,1),2)))} \\
& \sim_A & \underline{f(f(0,1),f(0,f(1,2)))} \\
& \rightarrow_{\mathcal{E}xt_{AC}(R),AC} & \cdots
\end{array}
$$

Example 84 shows that an essential property of minimal terms when considered as part of infinite $(\mathcal{E}xt_E(R),E)$-rewriting sequences for $A \vee C$-theories E gets lost: the application of $(\mathcal{E}xt_E(R),E)$-rewrite steps *at the root* of a minimal term s by means of a rule $l \to r$ (i.e., $s \sim_{AC} \sigma(l) \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_E(R)} \sigma(r)$) does *not* guarantee that there is a *nonvariable subterm* v of the right-hand side r which is a prefix of the 'next' minimal term in the infinite sequence. The problem illustrated in Example 84 is due to the application of associative steps at the root of a minimal term.

Example 85

Term t in Example 84 can be rewritten at the root *only* by rule (9.2) of $\mathcal{E}xt_{AC}(R)$. We can apply this rule to t' in Example 84 (for instance) to obtain $s' = \sigma(r) = f(f(0,f(1,2)),f(1,f(1,2)))$ (where $r = f(f(0,f(1,2)),y)$), which is $(\mathcal{E}xt_{AC}(R),AC)$-nonterminating. Note that s' contains a minimal term $u \in \mathcal{T}_{\infty,R,E}$. Since $s'|_2 = f(1,f(1,2))$ is $(\mathcal{E}xt_{AC}(R),AC)$-nonterminating, it follows that s' is *not* minimal. Since $s'|_1 = f(0,f(1,2))$ is $(\mathcal{E}xt_{AC}(R),AC)$-terminating, the only possibility is that u occurs in $s'|_2$. Actually, $s'|_2$ is minimal already; hence, $u = s'|_2$. But note the absence of any nonvariable position $p \in \mathcal{P}os(r)$ in the right-hand side of the considered rule such that $\sigma(r|_p) = u = f(1,f(1,2))$.

This is in sharp contrast with the situation of the DP approach for ordinary rewriting. Furthermore, it is not difficult to see that for all $t'' \sim_{AC} t$ such that $t'' = \sigma'(l)$ for some substitution σ', we have a similar situation. Thus, the problem illustrated here cannot be solved by using a different $\sim_{AC}$ sequence before performing the $\mathcal{E}xt_{AC}(R)$-root-step.

In the following we introduce a new notion of minimality which solves these problems.

9.3 A New Notion of Minimal E-Nonterminating Terms

The following definition solves the problems discussed above by explicitly requiring that the condition defining minimality is preserved under E-equivalence.

Definition 86 (Stably Minimal E-Nonterminating Term [ALM10]) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. Let $\mathcal{M}_{\infty,R,E}$ be the set of stably minimal E-nonterminating terms in the following sense: $t \in \mathcal{T}(\Sigma, \mathcal{X})$ belongs to $\mathcal{M}_{\infty,R,E}$ iff t is E-nonterminating, and for all $t' \sim_E t$ and every proper subterm s' of t' (i.e., $t' \rhd s'$), s' is $(\mathcal{E}xt_E(R), E)$-terminating.*

We have the following useful characterization of minimality.

Proposition 87 (Characterization of Stably Minimal Terms [ALM10]) *Let $\mathcal{R} = (\Sigma, R, E)$ be a rewrite theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, $t \in \mathcal{M}_{\infty,R,E}$ if and only if $[t]_E \subseteq \mathcal{T}_{\infty,R,E}$. Therefore,*

$$\mathcal{M}_{\infty,R,E} = \{t \in \mathcal{T}(\Sigma, \mathcal{X}) \mid [t]_E \subseteq \mathcal{T}_{\infty,R,E}\}$$

The problem in Example 84 disappears now: t is *not* (stably) minimal according to Definition 86. The same situation happens with the problem in Example83: $f(f(1,0),0) \in \mathcal{T}_{\infty,R,E}$ but $f(f(1,0),0) \notin \mathcal{M}_{\infty,R,E}$ since $f(f(1,0),0) \sim_E f(f(0,0),1)$ and $f(0,0)$ is E-nonterminating. In fact, $f(0,0) \in \mathcal{M}_{\infty,R,E}$.

The following result shows how to *find* stably minimal E-nonterminating terms associated to a given E-nonterminating term. This is essential in our development. A set of equations E is *size-preserving* if and only if for each equation $u = v$ the length of u and v are the same, i.e. $|u| = |v|$ and the multiset of the variables in u coincides with the multiset of the variables in v [Ohl02].

Proposition 88 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory such that E is regular and size-preserving. Let $s \in \mathcal{T}(\Sigma, \mathcal{X})$. If s is E-nonterminating, then there is a subterm t of some $s' \sim_E s$ (i.e., $s' \trianglerighteq t$) such that $t \in \mathcal{M}_{\infty,R,E}$.*

Clearly, Proposition 88 holds whenever $\mathcal{R}$ is an $A\vee C$-rewrite theory.

Example 89

Consider the term t in Example 84. Although $t \in \mathcal{T}_{\infty,R,E}$, $t \notin \mathcal{M}_{\infty,R,E}$: the term $t' = f(f(0,0), f(1, f(1,2)))$, which is AC-equivalent to t, contains a subterm $u = f(1, f(1,2))$ which is E-nonterminating. It is not difficult to see that actually $u \in \mathcal{M}_{\infty,R,E}$.

In general, Proposition 88 does *not* hold for arbitrary sets of equations E.

Example 90

Consider the following example [GK01, Example 13]:

$$R : f(x) \quad \to \quad x \qquad\qquad E : f(a) \quad = \quad a$$

Note that $a \in \mathcal{T}_{\infty,R,E}$. However, a is *not* stably minimal because $a \sim_E f(a)$ but $f(a) \notin \mathcal{T}_{\infty,R,E}$. Thus, Proposition 88 does not hold.

Since $\mathcal{M}_{\infty,R,E} \subseteq \mathcal{T}_{\infty,R,E}$, for $A\vee C$-rewrite theories E we have the following corollary.

Corollary 91 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory and $s \in \mathcal{M}_{\infty,R,E}$. If $s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} t$ and t is E-nonterminating, then $t \in \mathcal{T}_{\infty,R,E}$.*

In general, Corollary 91 does *not* hold if we require that $t \in \mathcal{M}_{\infty,R,E}$.

Example 92

Term $u = f(f(1,1),2)$ in Example 85 is stably minimal: $u \in \mathcal{M}_{\infty,R,E}$. We have that $f(\underline{f(1,1)},2) \xrightarrow{>\Lambda}_R f(f(0, f(1,2)),2)$. Note that $f(f(0, f(1,2)),2) \notin \mathcal{M}_{\infty,R,E}$. We have

$$\underline{f(f(0, f(1,2)),2)} \sim_A f(0, \underline{f(f(1,2),2)}) \sim_A f(0, f(1, f(2,2)))$$

where $f(0, f(1, f(2,2)))$ contains a subterm $f(1, f(2,2))$ which is $(\mathcal{E}xt_E(R), E)$-nonterminating.

The problem arises when $s \in \mathcal{M}_{\infty,R,E}$ is such that $root(s)$ includes *associativity* among its axioms, that is, $A_f \in E_f$.
Now we provide a more precise result about where we can find stably minimal subterms within an E-nonterminating term for $A\vee C$-rewrite theories $\mathcal{R} = (\Sigma, E, R)$. In the following theorem, given a term s and a symbol f, by an *f-subterm* t of s (written $s \trianglerighteq_f t$) we mean a subterm t of s such that $t = s|_p$ and for all $q < p$, $root(s|_q) = f$. We also write $s \triangleright_f t$ if $s \trianglerighteq_f t$ and $s \neq t$.

Theorem 93 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. If s is E-nonterminating, then there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s ($s \trianglerighteq t$) and*

1. *If (1) $A_{root(t)} \notin E_{root(t)}$ or (2) $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then $t \in \mathcal{M}_{\infty,R,E}$.*
2. *If $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and $t \notin \mathcal{M}_{\infty,R,E}$, then there is $s' \sim_E t$ and a strict f-subterm u of s' (i.e., $s' \triangleright_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

The following result is just a convenient reformulation of the previous one.

Corollary 94 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. If s is E-nonterminating, then either there is a subterm $t \in \mathcal{M}_{\infty,R,E}$ of s ($s \trianglerighteq t$), or there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s satisfying that $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and such that there is $s' \sim_E t$ and a strict f-subterm u of s' ($s' \triangleright_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

9.4 Structure of (Stably) Minimal Infinite $A\vee C$-Rewrite Sequences

Now we analyze $A\vee C$-rewrite sequences starting from stably minimal $A\vee C$-nonterminating terms. First we consider a restricted case.

Proposition 95 *[ALM10] Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A\vee C$-rewrite theory. Let $s \in \mathcal{M}_{\infty,R,E}$ be such that $f = root(s)$ and either (1) $A_f \notin E_f$, or (2) $s = f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1)$, $root(s_2) \in \mathcal{C}$. Assume that for all $l \to r \in R$ such that $root(l) = f$ and all subterms v of r ($r \trianglerighteq v$) such that $v = g(v_1, v_2)$ for some associative symbol g, we have that $root(v_1), root(v_2) \notin \mathcal{X} \cup \{g\}$. Then, there exist $l \to r \in R$, a substitution σ and terms $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and $u \in \mathcal{M}_{\infty,R,E}$ such that*

$$s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} t \sim_E \sigma(l) \xrightarrow{\Lambda}_R \sigma(r) \trianglerighteq u$$

and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.

Unfortunately, stable minimality of (arbitrary) E-nonterminating terms s for $A\vee C$-theories E is not preserved under inner $(\mathcal{E}xt_E(R), E)$-rewritings (see Example 92). The problem arises when s is rewritten into a term like, e.g., $t = f(f(t_1, t_2), t_3)$ on which associative steps can be issued to rearrange t and possibly introducing an E-nonterminating term below the root, thus *losing* stable minimality.

However, as a consequence of previous results, the following theorem establishes the desired property for stable minimal $A\vee C$-nonterminating terms.

Theorem 96 *[AGLM11] Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. For all $s \in \mathcal{M}_{\infty,R,E}$, there exist $l \to r \in \mathcal{E}xt_E(R)$ and a substitution σ such that*

$$s \xrightarrow{>\Lambda}{}^{*}_{\mathcal{E}xt_E(R),E} t \sim_E t' \trianglerighteq_f t'' \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$$

$t'' \in \mathcal{M}_{\infty,R,E}$ and there is a nonvariable subterm v of r $(r \trianglerighteq v)$, such that either

1. *$v = f(v_1, v_2)$ for some associative symbol f, $root(v_1) \in \mathcal{X} \cup \{f\}$ or $root(v_2) \in \mathcal{X} \cup \{f\}$, $root(\sigma(v_1)) = f$ or $root(\sigma(v_2)) = f$, $\sigma(v) \in \mathcal{T}_{\infty,R,E}$ and there is a term $t' \sim_E \sigma(v)$ containing a strict f-subterm $u = f(u_1, u_2)$ $(t' \rhd_f u)$ such that $u \in \mathcal{M}_{\infty,R,E}$, or*
2. *$\sigma(v) \in \mathcal{M}_{\infty,R,E}$ otherwise.*

Example 84 shows that Theorem 96 does not hold for Giesl and Kapur's minimal terms $s \in \mathcal{T}_{\infty,R,E}$.

10

$A\vee C$-Dependency Pairs and Chains

Propositions 88 and 95 together with Theorems 93 and 96 are the basis for our definition of *$A\vee C$-dependency pairs* and the corresponding *chains*. Together, they show that given an $A\vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$, every E-nonterminating term s has an associated infinite $(\mathcal{E}xt_E(R), E)$-rewrite sequence starting from a stably minimal subterm $t \in \mathcal{M}_{\infty,R,E}$. Such a sequence proceeds as described in Proposition 95 and Theorem 96, depending on the shape of t.

This process is abstracted in the following definition of $A\vee C$-dependency pairs (Definition 97) and in the definition of chain below (Definition 99).

Definition 97 ($A\vee C$-Dependency Pairs [AGLM11]) *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A\vee C$-rewrite theory. Then, $\mathsf{DP}_E(R) = \{l^\sharp \to s^\sharp \mid l \to r \in \mathcal{E}xt_E(R), r \trianglerighteq s, root(s) \in \mathcal{D}, l \not\trianglerighteq v \sim_E s\}$ is the set of $A\vee C$-dependency pairs ($A\vee C$-DPs) of $\mathcal{R}$.*

Requiring $l \not\trianglerighteq v \sim_E s$ for $\mathsf{DP}_E(\mathcal{R})$ in Definition 97 follows Dershowitz's criteria [Der04] extended to $A\vee C$-rewrite theories. In general, due to the use of Dershowitz's criteria, the set of $A\vee C$-DPs which is obtained from Definition 97 is a subset of those which are obtained by particularizing Giesl and Kapur's definitions to the $A\vee C$ case.

Example 98

Consider the $A\vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 84. The set $\mathsf{DP}_E(R)$ consists of the following pairs:

$$F(x, x) \to F(0, f(1, 2)) \tag{10.1}$$
$$F(x, x) \to F(1, 2) \tag{10.2}$$

$$F(f(x,x),y) \to F(f(0,f(1,2)),y) \quad (10.3)$$
$$F(f(x,x),y) \to F(0,f(1,2)) \quad (10.4)$$
$$F(f(x,x),y) \to F(1,2) \quad (10.5)$$

If we want to *characterize* termination of $A\vee C$-rewrite theories as the absence of infinite (minimal) *chains of $A\vee C$-dependency pairs*, we have to introduce a suitable notion of chain which can be used with $A\vee C$-DPs. As in the DP framework, where the origin of *pairs* does not matter, we should rather think of another rewrite theory $\mathcal{P} = (\mathcal{G}, F, P)$ which is used together with $\mathcal{R}$ to build the chains.

In the following, given sets of equations E and F, we let $\doteq_{F,E} = (\overset{\Lambda}{\vdash\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\dashv}_E)^*$. Moreover, we define $\xrightarrow{\Lambda}{}^*_{\mathcal{S}_f}$ as the application of rules $l \to r \in \mathcal{S}$ such that $root(l) = f$.

Definition 99 ((Minimal) $A\vee C$-Chain of Pairs [AGLM11]) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS. An (F, P, E, R, S)-chain is a finite or infinite sequence of pairs $u_i \to v_i \in P$, together with substitutions σ and θ_i satisfying that, for all $i \geq 1$:*

1. *If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F$ or $v'_i = u'_i \in F$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then*

$$\sigma(v_i) \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \doteq_{F,E} \sigma(u_{i+1})$$

2. *and $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \sigma(u_{i+1})$, otherwise.*

An (F, P, E, R, S)-chain is called minimal *if for all $i \geq 1$, and $t'_i \doteq_{F,E} t_i$, t'_i is $(\mathcal{E}xt_E(R), E)$-terminating.*

As usual, in Definition 99 we assume that different occurrences of dependency pairs do not share any variable (renaming substitutions are used if necessary). Note that the definition derives directly from Theorem 96: First we have to look for the minimal term of $\sigma(v_i)$, i.e. t_i, (see Theorem 93) which can be rewritten by using $\xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E}$ and again, since minimality can be lost we have to apply again Theorem 93 to connect with the next pair in the chain. This more abstract notion of chain can be particularized to be used with $A\vee C$-DPs, by just taking

1. $P = \mathsf{DP}_E(R)$,
2. $F = E^\sharp$, where $E^\sharp = \{s^\sharp = t^\sharp \mid s = t \in E\}$, and
3. $\mathcal{S} = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$.

The following propositions interconnect the remaining details between infinite $A\vee C$-rewrite sequences starting by stable minimal $A\vee C$-nonterminating terms as shown in Theorem 96 and infinite minimal $A\vee C$-chains in Definition 99.

Proposition 100 *[AGLM11] Let Σ be a signature and E be a set of noncollapsing equations over Σ. Let $s,t \in \mathcal{T}(\Sigma,\mathcal{X})$. Then, $s \sim_E t$ if and only if $s^\sharp \mathrel{\dot{\simeq}}_{E^\sharp,E} t^\sharp$.*

Proposition 101 *[AGLM11] Let Σ be a signature, $f \in \Sigma$, and $s,t \in \mathcal{T}(\Sigma,\mathcal{X})$. Then, $s \trianglerighteq_f t$ if and only if $s^\sharp \xrightarrow{\Lambda}{}^*_{\mathcal{S}_f} t^\sharp$.*

Now, we can give the desired result.

Theorem 102 (Characterization of $A\vee C$-Termination [AL11, ALM10]) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. Then, $\mathcal{R}$ is $(\mathcal{E}xt_E(R), E)$-terminating if and only if there is no infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain.*

11

$A\vee C$-Dependency Pair Framework

In the following, we extend Giesl et al.'s DP framework to provide a suitable framework for mechanizing proofs of $A\vee C$-termination using $A\vee C$-DPs as we have made in Part I for innermost *CSR*.

Definition 103 ($A\vee C$ Problem [ALM10]) *An $A\vee C$ problem τ is a tuple $\tau = (F, P, E, R, S)$, where $\mathcal{R} = (\Sigma, E, R)$ is an $A\vee C$-rewrite theory, $\mathcal{P} = (\mathcal{G}, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ is a TRS. An $A\vee C$ problem is finite if there is no infinite minimal (F, P, E, R, S)-chain. An $A\vee C$ problem τ is infinite if $\mathcal{R}$ is E-nonterminating or there is an infinite minimal (F, P, E, R, S)-chain.*

The following definition extends the notion of *DP processor* to prove termination of $A\vee C$-rewrite theories.

Definition 104 ($A\vee C$ Processor [ALM10]) *An $A\vee C$ processor* Proc *is a mapping from $A\vee C$ problems into sets of $A\vee C$ problems. Alternatively, it can also return* "no"*. An $A\vee C$ processor* Proc *is*

- sound *if for all $A\vee C$ problems τ, τ is finite whenever* $\mathsf{Proc}(\tau) \neq \mathsf{no}$ *and* $\forall \tau' \in \mathsf{Proc}(\tau)$*, τ' is finite.*
- complete *if for all $A\vee C$ problems τ, τ is infinite whenever* $\mathsf{Proc}(\tau) = \mathsf{no}$ *or* $\exists \tau' \in \mathsf{Proc}(\tau)$ *such that τ' is infinite.*

Similar to [GTSF06] for the DP framework, we construct a tree whose nodes are labeled with $A\vee C$ problems or *"*yes*"* or *"*no*"*, and whose root is labeled with $(E^\sharp, \mathsf{DP}_E(R), E, R, S)$. Now we have the following result which extends [GTSF06, Corollary 5] to $A\vee C$-rewrite theories.

Theorem 105 ($A \vee C$-DP Framework [ALM10]) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. We construct a tree whose nodes are labeled with $A \vee C$ problems or* "yes" *or* "no"*, and whose root is labeled with $(E^\sharp, \mathsf{DP}_E(R), E, R, S)$, where $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. For every inner node labeled with τ, there is a sound processor* Proc *satisfying one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child, labeled with* "no".
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child, labeled with* "yes".
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$, *and the children of the node are labeled with the $A \vee C$ problems in* $\mathsf{Proc}(\tau)$.

If all leaves of the tree are labeled with "yes"*, then $\mathcal{R}$ is E-terminating. Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is not E-terminating.*

12

$A \vee C$ Processors

In this chapter, we present several techniques that we have developed to deal with proofs of $A \vee C$-termination in the $A \vee C$-DP framework.

12.1 Preprocessing

A simple technique that can be useful when dealing with proofs of termination in the DP framework is to try to remove rules from the original system before building the DP problem. In this way, we will start the proof with less rules and therefore less pairs, which can simplify the proof of termination. We extend its use for proving E-termination. Here, $\sim$ is the stable, reflexive, transitive, and symmetric equivalence induced by $\gtrsim$, i.e., $\sim = \gtrsim \cap \lesssim$.

Proposition 106 (Removing strict rewrite rules [AGLM11]) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. Let $(\gtrsim, \sqsupset)$ be a monotonic reduction pair such that $l \; (\gtrsim \cup \sqsupset) \; r$ for all $l \to r \in R$ and $u \sim v$ for all $u = v \in E$. Let $R_{\sqsupset} = \{l \to r \in R \mid l \sqsupset r\}$ and $R' = R - R_{\sqsupset}$. Then, $\mathcal{R}$ is E-terminating if and only if $\mathcal{R}' = (\Sigma, E, R')$ is E-terminating.*

Example 107

Consider the following $A \vee C$-rewrite theory:

$$\mathtt{f}(\mathtt{g}(\mathtt{f}(\mathtt{h}(\mathtt{a}), \mathtt{a})), \mathtt{a}) \quad \to \quad \mathtt{f}(\mathtt{h}(\mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{a}))$$

where $\mathtt{f} \in \Sigma_{AC}$.

We can find a monotonic reduction pair such that $\mathtt{f}(\mathtt{g}(\mathtt{f}(\mathtt{h}(\mathtt{a}), \mathtt{a})), \mathtt{a}) \sqsupset \mathtt{f}(\mathtt{h}(\mathtt{a}), \mathtt{f}(\mathtt{a}, \mathtt{a}))$ and $u \sim v$ for all $u = v \in A \vee C$ with the following polynomial interpretation:

$$[\mathtt{f}](x, y) = x + y \qquad [\mathtt{a}] = 0 \qquad [\mathtt{g}](x) = x + 2 \qquad [\mathtt{h}](x) = x + 2$$

Therefore, we can remove the only rule in the system and therefore, conclude trivially its $A\vee C$-termination since there are no pairs and therefore, no infinite minimal $A\vee C$-chain.

After presenting this technique for simplifying proofs of $A\vee C$-termination, we introduce specific $A\vee C$ processors.

12.2 $A\vee C$-Dependency Graph

$A\vee C$ problems focus our attention on the analysis of *infinite minimal* $A\vee C$-*chains*. Our aim here is to obtain a notion of graph which is able to represent all infinite *minimal* $A\vee C$-chains of pairs as given in Definition 99.

Definition 108 ($A\vee C$-Graph of Pairs [ALM10]) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS. The $A\vee C$-*graph *associated to them (denoted $\mathsf{G}(F, P, E, R, S)$) has P as the set of nodes, and there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $u \to v, u' \to v'$ is an (F, P, E, R, S)-chain.*

Now we can use these notions to introduce the $A\vee C$-dependency graph, i.e., the $A\vee C$-graph whose nodes are the $A\vee C$-DPs instead of an arbitrary set of pairs.

Definition 109 ($A\vee C$-Dependency Graph [ALM10]) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that $S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. The $A\vee C$-*dependency graph *(A$\vee$C-DG) associated to $\mathcal{R}$ is:*

$$\mathsf{DG}(\mathcal{R}) = \mathsf{G}(E^\sharp, \mathsf{DP}_E(R), E, R, S)$$

12.3 Estimating the $A\vee C$-Dependency Graph

As in standard rewriting, the $A\vee C$-dependency graph of an $A\vee C$-rewrite theory is in general *not* computable. So, we need to use some approximation of it. For any term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and set Δ (which is intended to represent the set of defined symbols in $\mathcal{R}$) let $\textsc{Cap}_\Delta(t)$ result from replacing all subterms in t which are rooted by a symbol in Δ by fresh variables. Given a term t, we let $\textsc{Ren}(t)$ be the term which is obtained by *independently* renaming all *occurrences* of variables in t by using fresh variables [AG00].

As usual, we should not talk about an *mgu* when dealing with rewriting modulo equations. Instead, the appropriate notion is that of complete set of E-unifiers. The set of all E-unifiers of a given E-unifIcation problem is usually infinite. For equational unification, a single E-unifier is not always sufficient to represent all unifiers [BN98]. Fortunately, for approximating *reachability* we only need to check the *existence* of one.

Now, we are ready to provide a correct estimation of our graph of pairs.

Definition 110 (Estimated $A\vee C$-Graph of Pairs) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS. The* estimated $A\vee C$-graph *associated to them (denoted $\mathsf{EG}(F, P, E, R, S)$) has P as the set of nodes and arcs which connect them as follows:*

1. *If v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $root(u') = f$.*

2. *Otherwise, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $\textsc{Ren}(\textsc{Cap}_\Delta(v))$ and u' $(F \cup E)$-unify (where equations in F can only be applied at root position).*

According to Definition 108, we would have the following definition of *estimated* $A\vee C$-DG: $\mathsf{EDG}(\mathcal{R}) = \mathsf{EG}(E^\sharp, \mathsf{DP}_E(R), E, R, S)$, where again

$$S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

In the following result, given two sets of rules S and Q, we let S_Q be the least subset of S satisfying that whenever there is a rule $u \to v \in Q$, such that v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then $S_f \subseteq S_Q$.

Theorem 111 (SCC Processor [ALM10]) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS. Then, the processor Proc_{SCC} given by*

$$\mathsf{Proc}_{SCC}(F, P, E, R, S) = \{(F, Q, E, R, S_Q) \mid Q \text{ are the pairs of an SCC in } \mathsf{EG}(F, P, E, R, S)\}$$

is sound and complete.

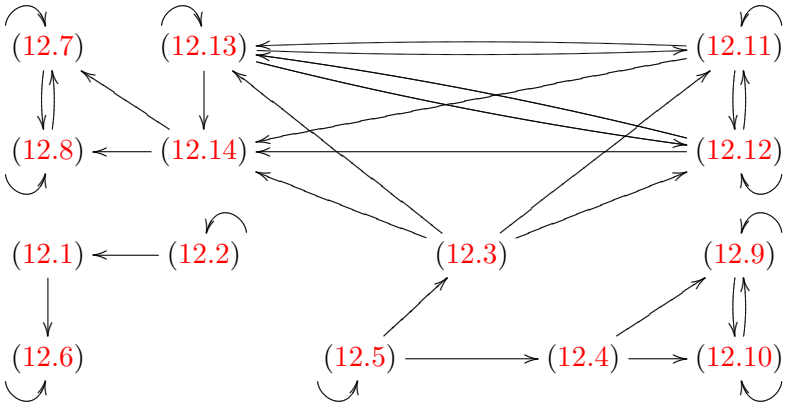
As a consequence, we can *separately* work with the SCCs of $\mathsf{EG}(F, P, E, R, S)$, disregarding other parts of the graph.

Example 112

For the $A\vee C$-rewrite theory in Example 9 (see page 21), the set $\mathsf{DP}_E(R)$ is[1]:

$$\begin{aligned}
\mathtt{LIST2SET}(\mathtt{cons}(N,L)) &\to \mathtt{UNION}(N,\mathtt{list2set}(L)) && (12.1)\\
\mathtt{LIST2SET}(\mathtt{cons}(N,L)) &\to \mathtt{LIST2SET}(L) && (12.2)\\
\mathtt{IN}(N,\mathtt{union}(M,S)) &\to \mathtt{EQ}(N,M) && (12.3)\\
\mathtt{IN}(N,\mathtt{union}(M,S)) &\to \mathtt{OR}(\mathtt{eq}(N,M),\mathtt{in}(N,S)) && (12.4)\\
\mathtt{IN}(N,\mathtt{union}(M,S)) &\to \mathtt{IN}(N,S) && (12.5)\\
\mathtt{UNION}(\mathtt{union}(N,N),Z) &\to \mathtt{UNION}(N,Z) && (12.6)\\
\mathtt{AND}(\mathtt{and}(\mathtt{true},B),Z) &\to \mathtt{AND}(B,Z) && (12.7)\\
\mathtt{AND}(\mathtt{and}(\mathtt{false},B),Z) &\to \mathtt{AND}(\mathtt{false},Z) && (12.8)\\
\mathtt{OR}(\mathtt{or}(\mathtt{true},B),Z) &\to \mathtt{OR}(\mathtt{true},Z) && (12.9)\\
\mathtt{OR}(\mathtt{or}(\mathtt{false},B),Z) &\to \mathtt{OR}(B,Z) && (12.10)\\
\mathtt{EQ}(\mathtt{s}(N),\mathtt{s}(M)) &\to \mathtt{EQ}(N,M) && (12.11)\\
\mathtt{EQ}(\mathtt{cons}(N,L),\mathtt{cons}(M,L')) &\to \mathtt{EQ}(N,M) && (12.12)\\
\mathtt{EQ}(\mathtt{cons}(N,L),\mathtt{cons}(M,L')) &\to \mathtt{EQ}(L,L') && (12.13)\\
\mathtt{EQ}(\mathtt{cons}(N,L),\mathtt{cons}(M,L')) &\to \mathtt{AND}(\mathtt{eq}(N,M),\mathtt{eq}(L,L')) && (12.14)
\end{aligned}$$

The (estimated) $A\vee C$-DG is:



By using Theorem 111 we transform the $A\vee C$ problem $(E^\sharp, \mathsf{DP}(R), E, R, S)$

[1] We have introduced new 'prefix' symbols *eq*, *cons*, and *union* instead of the original 'infix' ones _==_, _;_, __.

into a set of $A \vee C$ problems $\mathsf{Proc}_{SCC}(E^\sharp, \mathsf{DP}(R), E, R, S)$ given by

$\{(E^\sharp, \{(12.2)\}, E, R, \varnothing), (E^\sharp, \{(12.5)\}, E, R, \varnothing), (E^\sharp, \{(12.6)\}, E, R, S_{union}),$
$(E^\sharp, \{(12.7), (12.8)\}, E, R, S_{and}), (E^\sharp, \{(12.9), (12.10)\}, E, R, S_{or}),$
$(E^\sharp, \{(12.11), (12.12), (12.13)\}, E, R, \varnothing)\}$

which contains six new (but simpler) $A \vee C$ problems.

12.4 F Usable Equations Processor

Many times, the set of F axioms can be reduced to those equations that are really involved in minimal $A \vee C$-chains. The following processor shows a trivial method to eliminate them.

Theorem 113 (F Usable Equations Processor [AGLM11]) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS such that*

1. $root(u), root(v) \in \mathcal{G} - \Sigma$ *for all* $u \to v \in P$,
2. $root(s) = root(t) \in \mathcal{G} - \Sigma$ *for all* $s = t \in F$, *and*
3. $root(l) = root(r) \in \mathcal{G} - \Sigma$ *for all* $l \to r \in S$, *and*

Let $\hat{F} = \{s = t \in F \mid root(s) = root(u) \text{ or } root(s) = root(v) \text{ for some } u \to v \in P\}$

Then, the processor Proc_{FUEq} *given by*

$$\mathsf{Proc}_{FUEq}(F, P, E, R, S) = \{(\hat{F}, P, E, R, S)\}$$

is sound and complete.

Example 114

With respect to Example 112, we have $\tau_0 = (E^\sharp, \mathsf{DP}(R), E, R, S)$. By applying the SCC processor, we obtain $\mathsf{Proc}_{SCC}(\tau_0) = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$, where

- $\tau_1 = (E^\sharp, \{(12.2)\}, E, R, \varnothing)$,
- $\tau_2 = (E^\sharp, \{(12.5)\}, E, R, \varnothing)$,
- $\tau_3 = (E^\sharp, \{(12.6)\}, E, R, S_{union})$,

- $\tau_4 = (E^\sharp, \{(12.7), (12.8)\}, E, R, S_{and})$,
- $\tau_5 = (E^\sharp, \{(12.9), (12.10)\}, E, R, S_{or})$ and
- $\tau_6 = (E^\sharp, \{(12.11), (12.12), (12.13)\}, E, R, \varnothing)$.

We can apply Proc_{FUEq} to all these $A \vee C$ problems:

- For τ_1, we have $\mathsf{Proc}_{FUEq}(\tau_1) = (\varnothing, \{(12.2)\}, E, R, \varnothing)$,
- For τ_2, we have $\mathsf{Proc}_{FUEq}(\tau_2) = (\varnothing, \{(12.5)\}, E, R, \varnothing)$,
- For τ_3, we have $\mathsf{Proc}_{FUEq}(\tau_3) = (E^\sharp_{union}, \{(12.6)\}, E, R, S_{union})$,
- For τ_4, we have $\mathsf{Proc}_{FUEq}(\tau_4) = (E^\sharp_{and}, \{(12.7), (12.8)\}, E, R, S_{and})$,
- For τ_5, we have $\mathsf{Proc}_{FUEq}(\tau_5) = (E^\sharp_{or}, \{(12.9), (12.10)\}, E, R, S_{or})$ and
- For τ_6, we have $\mathsf{Proc}_{FUEq}(\tau_6) = (E^\sharp_{eq}, \{(12.11), (12.12), (12.13)\}, E, R, \varnothing)$.

12.5 Use of Reduction Pairs

In the dependency pair framework reduction pairs are used to obtain *smaller* sets of pairs $\mathcal{P}' \subseteq \mathcal{P}$ by removing the *strict* pairs, i.e., those pairs $u \to v \in \mathcal{P}$ such that $u \sqsupset v$. Dealing with associative and/or commutative axioms, we will compare them with the equivalence relation defined by the stable, reflexive, transitive, and symmetric equivalence $\sim$ induced by $\gtrsim$, i.e., $\sim = \gtrsim \cap \lesssim$, since we need to impose compatibility with the equational theories E and F. The following theorem formalizes a generic processor to remove pairs from $\mathcal{P}$ by using reduction pairs.

Theorem 115 (Reduction Pair Processor [AGLM11]) *Let $\mathcal{P} = (\mathcal{G}, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ be a TRS. Let $(\gtrsim, \sqsupset)$ be a reduction pair such that*

1. *$R \subseteq \gtrsim$,*
2. *$P \cup S \subseteq \gtrsim \cup \sqsupset$, and*
3. *$E \cup F \subseteq \sim$.*

Let $P_{\sqsupset} = \{u \to v \in P \mid u \sqsupset v\}$ *and* $S_{\sqsupset} = \{s \to t \in S \mid s \sqsupset t\}$. *Then, the processor* Proc_{RP} *given by*

$$\mathsf{Proc}_{RP}(F,P,E,R,S) = \begin{cases} \{(F, P - P_{\sqsupset}, E, R, S - S_{\sqsupset})\} & \text{if (1), (2), and (3) hold} \\ \{(F,P,E,R,S)\} & \text{otherwise} \end{cases}$$

is sound and complete.

12.5.1 Usable Rules and Equations for $A \vee C$ Problems

Usable rules are widely used in the DP framework to improve the power of DP processors. We show how to obtain the set of usable rules and *usable equations* for a given $A \vee C$ problem and how to use them to define a new reduction pair processor. We follow the approach and techniques developed in [HM04, TGS04]. Our first intuition was to define a proper notion of dependency that not only takes into account the symbols occurring in the rules, but also the symbols occurring in the equations; but, since $A \vee C$ equations do not introduce new symbols in their left- and right-hand sides, we can use the standard notion of dependency that only considers symbols occurring in the rules. We use some auxiliary definitions. Let $Rls_R(f) = \{l \to r \in R \mid root(l) = f\}$, $Eqs_E(f) = \{u = v \in E \mid root(u) = f \vee root(v) = f\}$. Let $Fun(t) = \{f \mid \exists p \in \mathcal{P}os_{\mathcal{F}}(t), f = root(t|_p)\}$.

Definition 116 (Dependency [Urb04]) *Let* $\mathcal{R} = (\Sigma, R)$ *be a TRS. We say that* $f \in \Sigma$ *has a* dependency *on* $h \in \Sigma$ *(written* $f \rhd_R h$*) if* $f = h$ *or there is a function symbol* g *with* $g \rhd_R h$ *and a rule* $l \to r \in Rls_R(f)$ *with* $g \in Fun(r)$.

To obtain the correct notions of usable rule and equation, we have to look at the structure of the chains. We have two possible ways to proceed in an (F,P,E,R,S)-chain. Given $u_i \to v_i \in P$ either

$$\sigma(v_i) \doteqdot_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \;\to^*_{\mathcal{E}xt_E(R),E} \circ \doteqdot_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \doteqdot_{F,E} \sigma(u_{i+1})$$

or

$$\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteqdot_{F,E} \sigma(u_{i+1})$$

Then, to obtain the set of usable rules and also usable equations we have to look for usable symbols not only in P, but also in F and S.

Definition 117 ($A \vee C$-Usable Rules and Equations [AGLM11]) *Let* τ *be an* $A \vee C$ *problem such that* $\tau = (F,P,E,R,S)$ *where* $\mathcal{R} = (\Sigma, E, R)$ *is*

*an $A \vee C$-rewrite theory, $\mathcal{P} = (\mathcal{G}, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ is a TRS. The set $\mathcal{U}_R(\tau)$ of $A \vee C$-*usable rules *of τ is*

$$\begin{aligned}\mathcal{U}_R(\tau) \;=\; & \bigcup_{s \to t \,\in\, P, f \,\in\, Fun(t), f \triangleright_R g} Rls_R(g) \;\cup \\ & \bigcup_{u = v \,\in\, F,\; f \,\in\, Fun(u) \,\cup\, Fun(v), f \triangleright_R g} Rls_R(g) \;\cup \\ & \bigcup_{l \to r \,\in\, S,\; f \,\in\, Fun(r), f \triangleright_R g} Rls_R(g)\end{aligned}$$

*The set $\mathcal{U}_E(\tau)$ of $A \vee C$-*usable equations *of τ is*

$$\begin{aligned}\mathcal{U}_E(\tau) \;=\; & \bigcup_{s \to t \,\in\, P,\; f \,\in\, Fun(t), f \triangleright_R g} Eqs_E(g) \;\cup \\ & \bigcup_{u = v \,\in\, F,\; f \,\in\, Fun(u) \,\cup\, Fun(v), f \triangleright_R g} Eqs_E(g) \;\cup \\ & \bigcup_{l \to r \,\in\, S, f \,\in\, Fun(r), f \triangleright_R g} Eqs_E(g)\end{aligned}$$

Note that, if the rules from S are of the form $f^\sharp(f(x,y),z) \to f^\sharp(x,y)$ or $f^\sharp(x, f(y,z)) \to f^\sharp(y,z)$, then they do not introduce new rules as usable.

A TRS $\mathcal{R}$ is $\mathcal{C}_\varepsilon$-terminating if $\mathcal{R} \uplus \mathcal{C}_\varepsilon$ is terminating, where $\mathcal{C}_\varepsilon = \{\mathsf{c}(x,y) \to x,\ \mathsf{c}(x,y) \to y\}$ (and c is not in the signature of $\mathcal{R}$). A relation $\gtrsim$ is $\mathcal{C}_\varepsilon$-compatible iff $\mathsf{c}(x,y) \gtrsim x$ and $\mathsf{c}(x,y) \gtrsim y$. We need these notions to define the following important result.

Theorem 118 (RP Processor with $A \vee C$-Usable Rules and Equations [AGLM11]) *Let $\tau = (F, P, E, R, S)$ be an $A \vee C$ problem where $\mathcal{R} = (\Sigma, E, R)$ is an $A \vee C$-rewrite theory, $\mathcal{P} = (\mathcal{G}, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{H}, S)$ is a TRS. Let $(\gtrsim, \sqsupset)$ be a reduction pair such that $\gtrsim$ is $\mathcal{C}_\varepsilon$-compatible and*

1. *$\mathcal{U}_R(\tau) \subseteq \gtrsim$,*
2. *$P \cup S \subseteq \gtrsim \cup \sqsupset$, and*
3. *$F \cup \mathcal{U}_E(\tau) \subseteq \sim$.*

Let $P_\sqsupset = \{u \to v \in P \mid u \sqsupset v\}$ and $S_\sqsupset = \{s \to t \in S \mid s \sqsupset t\}$. Then, the processor Proc_{UR} given by

$$\mathsf{Proc}_{UR}(F,P,E,R,S) = \begin{cases} \{(F, P - P_\sqsupset, E, R, S - S_\sqsupset)\} & \text{if (1), (2), and (3) hold} \\ \{(F,P,E,R,S)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Example 119

For the $A \vee C$-rewrite theory in Example 9, we have the following rules in R (with prefix symbols again):

$$\begin{aligned}
\texttt{list2set}(N) &\to N && (12.15)\\
\texttt{list2set}(\texttt{cons}(N, L)) &\to \texttt{union}(N, \texttt{list2set}(L)) && (12.16)\\
\texttt{in}(N, \texttt{null}) &\to \texttt{false} && (12.17)\\
\texttt{in}(N, \texttt{union}(M, S)) &\to \texttt{or}(\texttt{eq}(N, M), \texttt{in}(N, S)) && (12.18)\\
\texttt{union}(N, N) &\to N && (12.19)\\
\texttt{and}(\texttt{true}, B) &\to B && (12.20)\\
\texttt{and}(\texttt{false}, B) &\to \texttt{false} && (12.21)\\
\texttt{or}(\texttt{true}, B) &\to \texttt{true} && (12.22)\\
\texttt{or}(\texttt{false}, B) &\to B && (12.23)\\
\texttt{eq}(0, \texttt{s}(N)) &\to \texttt{false} && (12.24)\\
\texttt{eq}(\texttt{s}(N), \texttt{s}(M)) &\to \texttt{eq}(N, M) && (12.25)\\
\texttt{eq}(\texttt{cons}(N, L), M) &\to \texttt{false} && (12.26)\\
\texttt{eq}(\texttt{cons}(N, L), \texttt{cons}(M, L')) &\to \texttt{and}(\texttt{eq}(N, M), \texttt{eq}(L, L')) && (12.27)\\
\texttt{eq}(L, L) &\to \texttt{true} && (12.28)
\end{aligned}$$

In Example 114, we had the following $A \vee C$ problems:

- $\tau_1' = (\varnothing, \{(12.2)\}, E, R, \varnothing)$,
- $\tau_2' = (\varnothing, \{(12.5)\}, E, R, \varnothing)$,
- $\tau_3' = (E^\sharp_{union}, \{(12.6)\}, E, R, S_{union})$,
- $\tau_4' = (E^\sharp_{and}, \{(12.7), (12.8)\}, E, R, S_{and})$,
- $\tau_5' = (E^\sharp_{or}, \{(12.9), (12.10)\}, E, R, S_{or})$ and
- $\tau_6' = (E^\sharp_{eq}, \{(12.11), (12.12), (12.13)\}, E, R, \varnothing)$.

We can apply Proc_{UR} to all these $A \vee C$ problems:

- In the case of τ_1' we have:
  $\mathcal{U}_R(\tau_1') = \varnothing$, $\mathcal{U}_E(\tau_1') = \varnothing$ and the following polynomial interpretation[2]

$$[\texttt{LIST2SET}](x) = x * x + x \qquad [\texttt{cons}](x, y) = y + 1$$

[2]The quasi-orders $\gtrsim$ induced by a polynomial interpretation can always be made $\mathcal{C}_\varepsilon$-compatible with the rules of the TRS $\mathcal{C}_\varepsilon$, i.e., $\mathcal{C}_\varepsilon \subseteq \gtrsim$.

to conclude finiteness of τ_1'.

- For τ_2' we have:
  $\mathcal{U}_R(\tau_2') = \varnothing$, $\mathcal{U}_E(\tau_2') = \varnothing$ and the following polynomial interpretation:

$$[\texttt{IN}](x,y) = x * y + y \qquad [\texttt{union}](x,y) = y + 1$$

  to conclude finiteness of τ_2'.

- For τ_3' we have:
  $\mathcal{U}_R(\tau_3') = \{(12.19)\}$, $\mathcal{U}_E(\tau_3') = \{(E_{union})\}$ and the following polynomial interpretation:

$$[\texttt{UNION}](x,y) = x + y + 1 \qquad [\texttt{union}](x,y) = x + y$$

  to conclude finiteness of τ_3'.

- For τ_4' we have:
  $\mathcal{U}_R(\tau_4') = \{(12.20), (12.21)\}$, $\mathcal{U}_E(\tau_4') = \{(E_{and})\}$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{AND}](x,y) & = & x + y & \qquad & [\texttt{and}](x,y) & = & x + y + 1 \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

  This processor eliminates one strict pair and generates a new $A \vee C$ problem $\tau_{4.1} = (E_{and}^{\sharp}, \{(12.7)\}, E, R, S_{and})$ where again we have:
  $\mathcal{U}_R(\tau_{4.1}) = \{(12.20), (12.21)\}$, $\mathcal{U}_E(\tau_{4.1}) = \{(E_{and})\}$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{AND}](x,y) & = & x + y & \qquad & [\texttt{and}](x,y) & = & x + y \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

  to conclude finiteness of $\tau_{4.1}$ and therefore of τ_4'.

- For τ_5' we have:
  $\mathcal{U}_R(\tau_5') = \{(12.22), (12.23)\}$, $\mathcal{U}_E(\tau_5') = \{(E_{or})\}$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{OR}](x,y) & = & x * y + x + y & \qquad & [\texttt{or}](x,y) & = & x * y + x + y \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

This processor eliminates one strict pair and generates a new $A \vee C$ problem $\tau_{5.1} = (E^{\sharp}_{or}, \{(12.9)\}, E, R, S_{or})$ where again we have:

$\mathcal{U}_R(\tau_{5.1}) = \{(12.22), (12.23)\}$, $\mathcal{U}_E(\tau_{5.1}) = \{(E_{and})\}$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{OR}](x,y) & = & x+y & & [\texttt{or}](x,y) & = & x+y+1 \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

to conclude finiteness of $\tau_{5.1}$ and therefore of τ'_5.

- Finally, for τ'_6 we have:
  $\mathcal{U}_R(\tau'_6) = \varnothing$, $\mathcal{U}_E(\tau'_6) = \varnothing$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{EQ}](x,y) & = & x*y+x+y & & [\texttt{cons}](x,y) & = & x+y+1 \\ [\texttt{s}](x) & = & x+1 & & & & \end{array}$$

  This processor eliminates one strict pair and generates a new $A \vee C$ problem $\tau_{6.1} = (E^{\sharp}_{eq}, \{(12.11), (12.12)\}, E, R, \varnothing)$ where we have:

  $\mathcal{U}_R(\tau_{6.1}) = \varnothing$, $\mathcal{U}_E(\tau_{6.1}) = \varnothing$ and the following polynomial interpretation:

$$\begin{array}{rclcrcl} [\texttt{EQ}](x,y) & = & x*y+x+y & & [\texttt{cons}](x,y) & = & x+1 \\ [\texttt{s}](x) & = & x+1 & & & & \end{array}$$

  This application eliminates another strict pair and generates a new $A \vee C$ problem $\tau_{6.2} = (E^{\sharp}_{eq}, \{(12.11)\}, E, R, \varnothing)$. We have:

  $\mathcal{U}_R(\tau_{6.2}) = \varnothing$, $\mathcal{U}_E(\tau_{6.2}) = \varnothing$ and the following polynomial interpretation:

$$[\texttt{EQ}](x,y) = x*y+x+y \qquad [\texttt{s}](x) = x+1$$

  to conclude finiteness of $\tau_{6.2}$ and therefore of τ'_6.

Therefore, after showing the finiteness of all the $A \vee C$ problems generated from Example 9 (see page 21), we can conclude its E-termination.

13

Experiments on $A \vee C$-Rewriting

We have implemented all the techniques described in this part in the termination tool MU-TERM [AGLN10, AGIL07, Luc04]. As we have stated, MU-TERM is a tool that can be used to verify a number of termination properties of (variants of) TRSs. It includes termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and, thanks to this new approach, termination of rewriting modulo specific axioms. With these new features implemented, MU-TERM has been able to participate in the International Competition of Termination Tools in the category of *TRS Equational* in 2010. This is not the first implementation for proving termination of rewriting modulo axioms: C*i*ME [CMMU03] is able to prove AC-termination of TRSs, and AProVE [GST06] is able to deal with termination of rewriting modulo equations satisfying some restrictions. However, in the latest editions of the competition, C*i*ME did not participate and AProVE is the only termination tool that has participated in this category from its first edition in 2004. The TPDB contains 71 examples in the equational category[1]. In the 2010 edition, there were only two participants: AProVE and MU-TERM. In the case of MU-TERM, it only implemented the techniques described in [ALM10]. The organization randomly selected a subset of 34 examples from the entire set. MU-TERM was able to solve 16 out of them whereas AProVE solved 24. We considered this result as a good one since only a few techniques had been implemented to deal with termination modulo axioms and AProVE has implemented specific techniques since 2004. These include an AC-recursive path order (RPO) with status (3 examples are solved with it) and processors based on usable rules (the remaining 5 examples are solved using them). There is no formal publication on any of these techniques. In the case of the AC-RPO, we suppose that they implement the master thesis of Stephan Falke [Fal04],

[1] We have used version 7.0.2 of the TPDB.

	MU-TERM [AGLM11]	AProVE	MU-TERM [ALM10]
YES score	59	57	39
YES average time	6.83 sec.	5.12 sec.	40.13 sec.

Table 13.1: Comparison in proofs of termination of $A \vee C$-rewrite theories

although [Rub02] was published before. Recently, we have found out that this work was adapted to the dependency pair framework in the master thesis of Christian Stein ([Ste06], in German and not publicly available). However, both papers are based on the notion of minimality presented in [GK01], which we have shown is not appropriate. In the case of processors for managing usable rules, it is essential to deal with a correct notion of minimality [HM04, TGS04].

Nowadays, after implementing the techniques described in [AGLM11], MU-TERM has been able to solve 59 examples out of 71, two examples more than AProVE[2]. For full details of proofs see:

http://zenon.dsic.upv.es/muterm/benchmarks/benchmarks-avc/benchmarks.html

In comparison with the implementation of the techniques developed in [ALM10], where MU-TERM was able to solve 39 examples[3], thanks to the new techniques, MU-TERM has currently become a powerful and competitive tool for proving termination of $A \vee C$-rewrite theories. The practical results are summarized in Table 13.1 and the complete set of benchmarks can be consulted in Appendix B (Chapter 20).

[2]See the 2008 edition of the termination competition where the entire set of examples from the category were considered.

[3]See http://www.dsic.upv.es/~balarcon/WRLA10/benchmarks.html

14

Related Work and Contributions

As remarked in the introduction, this is not the first work that tries to use dependency pairs for proving termination of rewriting modulo an equational theory, see [Fal04, GK01, KMU02, Kus00, KNT06, KT01, MU98, MU04, Ste06]. However, our work is, as far as we know, the first one that provides a satisfactory notion of minimal nonterminating term for an $A\vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ which can be used to provide a suitable definition of minimal chain of dependency pairs, which can in turn be used to characterize $A\vee C$-termination (Theorem 102). In order to substantiate this claim, consider the AC-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 84 again. The $A\vee C$-DPs for $\mathcal{R}$ are enumerated in Example 98. These dependency pairs coincide with the ones which would be computed by, e.g., [Fal04, GK01, KNT06, KT01, Ste06]. Remember that t in Example 84 is *minimal* in Giesl and Kapur's sense (Definition 81); and also according to [Fal04, Ste06], which inherit this notion. We should then be able to find an infinite *minimal* chain of DPs starting from $t^\sharp$. According to [Fal04, GK01, KNT06, KT01, Ste06], 'minimal' means that $\sigma(v_i)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all pairs $u_i \to v_i \in \mathsf{DP}_E(R)$ in the chain of dependency pairs induced by the substitution σ. However, this is *not* possible: the marked version $t^\sharp$ of t is $F(f(0,1), f(0, f(1,2)))$, which is an $(\mathcal{E}xt_E(R), E)$-terminating term. However, after some $E^\sharp \cup E$-equivalence steps (where $E^\sharp$ is applied only at root position), we would be able to apply one of the rules in $\mathsf{DP}_E(R)$. Note, however, that *no rule* $u \to v \in \mathsf{DP}_E(R)$ except (10.3) has a right-hand side v that can be rewritten (after instantiation into $\sigma(v)$) into an instance $\sigma(u')$ of the left-hand side u' of any other pair in $\mathsf{DP}_E(R)$ by means of $(\mathcal{E}xt_E(R), E^\sharp \cup E)$-rewriting steps. This means that only the dependency pair (10.3) could be used in any infinite minimal chain of dependency pairs starting from $t^\sharp$. But such a chain would start as follows:

$$F(f(0,1), f(0, f(1,2))) \doteq_{E^\sharp, E} F(f(0,0), f(1, f(1,2))) \to_{(10.3)} F(f(0, f(1,2)), f(1, f(1,2)))$$

where $F(f(0, f(1, 2)), f(1, f(1, 2)))$ as showed in Example 84, contains a subterm $f(1, f(1, 2))$, which is $(\mathcal{E}xt_E(R), E)$-nonterminating. Therefore, this chain of dependency pairs is *not* minimal. We conclude that, according to the notion of minimal chain in the aforementioned papers, *there is no minimal chain of pairs starting from* $t^\sharp$. This means that no *sound* approach to proving AC-termination on the basis of such notion of minimal chain is possible.

We have introduced the notion of *stably minimal term* (Definition 86) which overcomes these problems (Proposition 95 and Theorem 96) and leads to an appropriate characterization of $A\vee C$-termination as the absence of infinite minimal chains of $A\vee C$-DPs (Definitions 97 and 99, and Theorem 102).

Comparing our DPs with the ones in [KMU02] we would get the same set of pairs for its "unmarked version", which lose the essence of the dependency pairs since it does not lead with tuple symbols and therefore, monotonicity cannot be disregarded in the strict order. In the case of the two "marked versions" they affirm that these procedures obtain more dependency pairs than the unmarked version and therefore, more than our approach. Moreover, they allow tuple symbols not only at root position losing again, the common use of dependency pairs.

Furthermore, we note that [KNT06, KT01, KMU02] deal with *AC-rewrite theories* only, and that [GK01], which considers more general rewrite theories E including $A\vee C$-theories, does not cover our work in a second respect: when purely associative theories are considered (i.e., rewrite theories $\mathcal{R} = (\Sigma, E, R)$ such that $E_f \subseteq \{A_f\}$ for all $f \in \Sigma$); then Giesl and Kapur's technique requires the computation of *instances* of the rules in $\mathcal{E}xt_E(R)$ for which the computation of *all* the E-unifiers $uni_E(v, l)$ of v and l for the rules $l \to r$ in $\mathcal{E}xt_E(R)$ and equations $u = v \in E$ or $v = u \in E$ is required. It is well-known, however, that the E-unification problem for associative theories E is *infinitary*, which means that $uni_E(v, l)$ is not guaranteed to be finite, in general. In sharp contrast, we do not have to do that to deal with purely associative rewrite theories $\mathcal{R}$.

Our second main contribution is the formalization of an $A\vee C$-dependency pair framework (Definitions 103 and 104) which, on the basis of the previously developed theory, can be used to develop automatic tools for proving termination of $A\vee C$-rewrite theories (Theorem 105). Several important processors have been developed as well: the SCC processor (Theorem 111), the processor that restricts the set of F axioms (Theorem 113), the reduction pair processor (Theorem 115), and the reduction pair processor with usable rules and equations (Theorem 118). We have implemented the techniques described here in the termination tool MU-TERM and we have developed some benchmarks, showing that our $A\vee C$-DP Framework is currently the most powerful approach

for proving termination of $A \vee C$-rewrite theories. As we have stated, the implementation of the techniques in [ALM10] allowed us to participate in the termination competition in the equational category in the TPDB and, therefore, provide MU-TERM with the ability to prove termination modulo axioms. Moreover, thanks to the improvements in [AGLM11], MU-TERM is a powerful tool for proving termination of $A \vee C$-rewrite theories and, as far as we know, no tool is able to solve more examples from the equational category of the TPDB. Even though much work remains ahead in terms of further developing the new $A \vee C$-dependency pair framework, we also have to take into consideration that our framework is applicable to an even wider range of rewrite theories, which can be transformed into $A \vee C$-theories by nontermination-preserving transformations [DLM09b, DLM$^+$08, LM09]. Nevertheless, appropriate reduction orders that are well-suited for use in the reduction pair processor should be investigated. It would also be very useful to explore how the requirements on E can be relaxed to handle even more general sets of axioms. Following our approach, we could be able to manage every equational theory E such that E is regular and linear (to ensure the equivalence between $\rightarrow_{\mathcal{E}xt_E(R),E}$ and $\rightarrow_{R/E}$) and contains noncollapsing equations (and maybe, some additional requirement on root symbols). Obviously, this let aside the identity axiom, which is very used in Maude. However, there exists a transformation for dealing with it in [DLM09b]. Disproving termination of $A \vee C$-rewrite theories is also an interesting future work to be investigated within the framework.

15

Termination Tools for Maude programs

15.1 MU-TERM 5.0

MU-TERM was originally designed to prove termination of *CSR* [Luc04]. However, the latest version, MU-TERM 5.0 [AGLN10] includes much more than that. We haver worked hard in the last years not only to improve its ability to prove termination of *CSR* but also to verify a number of *other* termination properties of (variants of) TRSs.

In contrast to *transformational* approaches which translate termination problems into a classical termination problem for TRSs, we have developed specific techniques to deal with termination of *CSR*, innermost *CSR*, order-sorted rewriting and rewriting modulo specific axioms (associative or commutative) by using dependency pairs. Our benchmarks show that direct methods lead to simpler, faster and more successful proofs. Moreover, MU-TERM 5.0 has been rewritten to embrace the dependency pair framework which is specially well-suited for mechanizing proofs of termination.

MU-TERM 5.0 consists of 47 Haskell modules with more than 19000 lines of code. Details about the implementation can be found in [Gut10, Chapter 7].

A web-based interface and compiled versions in several platforms are available at the MU-TERM 5.0 web site.

http://zenon.dsic.upv.es/muterm/

Its use is really simple: the user introduces a program in the text box in TPDB or OBJ/Maude format or uploads a file in any of those formats. Then, fix a timeout (by default 5 seconds) and press the button 'Prove automatically". That is all. This allows inexpert users to prove automatically termination by means of the automatic option. This is very convenient for teaching purposes, for instance.

In the DP framework, a strategy is applied to an initial (*CSR*, innermost *CSR*, ...) problem and returns a proof tree. This proof tree is later evaluated following a tree evaluation strategy (normally, breadth-first search).

The strategy of MU-TERM 5.0 is defined by using the following combinators:

- **And**: sequential combination of two processors.
- **Or**: adds a decision point.
- **Iterate**: applies a strategy until a fix point is reached (success or a don't know node is reached).

With small differences depending on the particular kind of problem, we do the following:

1. We check the system for extra variables (at active positions) in the right-hand side of the rules.
2. We check whether the system is innermost equivalent. If it is true, then we transform the problem into an innermost one.
3. Then, we obtain the corresponding dependency pairs, obtaining a DP problem. And now, recursively:
   (a) Decision point between infinite processors and the SCC processor.
   (b) Subterm criterion processor.
   (c) Reduction pair (RP) processor with linear polynomials (LPoly) and coefficients in $N_2 = \{0, 1, 2\}$.
   (d) RP processor with LPoly and coefficients in $Q_2 = \{0, 1, 2, \frac{1}{2}\}$ and $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{1}{4}\}$ (in this order).
   (e) RP processor with simple mixed polynomials (SMPoly) and coefficients in N_2.
   (f) RP processor with SMPoly and rational coefficients in Q_2.
   (g) RP processor with 2-square matrices with entries in N_2 and Q_2.
   (h) Transformation processors (only twice to avoid nontermination of the strategy): instantiation, forward instantiation, and narrowing.
4. If the techniques above fail, then we use RPO (for standard rewriting or *CSR*).

The explanation of processors not given here can be found in [AGL10]. Note also that all processors are new with respect to MU-TERM 4.3 [AGIL07].

More details about these experimental results in all considered termination properties can be found here:

http://zenon.dsic.upv.es/muterm/benchmarks/index.html

Therefore, MU-TERM 5.0 is no more a tool for proving termination of *CSR* only: We can say now that it has evolved to become a powerful termination tool which is able to prove termination of a wide range of interesting properties of rewriting with important applications to prove termination of programs in sophisticated rewriting-based programming languages like Maude or OBJ* (as we have shown in this Thesis). In fact, apart from MTT, it is the only termination tool that accepts programs in OBJ/Maude syntax.

15.2 MTT: The Maude Termination Tool

The MTT [DLM08a] is a tool that checks the termination of Maude equational specifications. The last version of MTT can not only check membership equational theories (like previous versions) but also the termination of rewrite theories. There are still several restrictions in the Maude specification: built-ins cannot be used and attributes `owise` and *identity* elements are not supported. MTT takes Maude modules as inputs and tries to prove them terminating by using a number of existing termination tools as backends. As we have mentioned, proving termination of expressive equational programs that have the features of Maude is nontrivial, since some of these are not supported by standard termination methods and tools. Yet, the use of such features may be essential to ensure termination. MTT uses the theory transformations described in [DLM$^+$04, DLM$^+$08, LM09] and summarized in Section 1.2, to bridge the gap between expressive equational programs and conventional termination tools for term rewriting systems, which are used as backends. It can send the transformed termination problems to any tool that supports the TPDB syntax currently shared by many such tools. In previous versions, MTT was able to interact with C*i*ME, AProVE, and MU-TERM, but now other formal tools, such as TTT, Termptation, etc. can be considered as backend as well. The tool implementation distinguishes two parts:

- a reflective Maude specification implements the theory transformations described in [DLM$^+$04, DLM$^+$08, LM09], and
- a Java application connects Maude to some term rewriting termination tools and provides a graphical user interface.

	AProVE	MU-TERM
YES score	18/27	12/27

Table 15.1: Comparison in proofs of termination of transformed Maude programs

	MU-TERM 5.0	AProVE
YES score	20/27	18/27

Table 15.2: Comparison in proofs of termination of transformed Maude programs with MU-TERM 5.0

The Java application is in charge of sending the Maude specification provided by the user to Maude to perform transformations. Depending on the selections, one transformation or another will be accomplished. The resulting unsorted unconditional rewriting system obtained from these transformations is proved terminating by using the above-mentioned tools as backends.

A comparison over a set of Maude examples can be found in:

http://www.lcc.uma.es/~duran/MTT/mtt15/Samples/index.html

There, several Maude examples are transformed by using the transformations described in [DLM+04, DLM+08, LM09] and summarized in Section 1.2. This process generates 27 examples that are sent to AProVE and MU-TERM to prove their termination. The results showed in this web are summarized in Table 15.1.

However, these results correspond to a previous version of MU-TERM, where the treatment of $A \vee C$-rewrite theories was not developed. Now, taking into account the new features of MU-TERM 5.0, we have repeated the benchmarks and the results are much better.

Complete details can be found in

http://www.dsic.upv.es/~balarcon/Thesis/MTT/benchmarks.html

and in Appendix C (Chapter 21). A summary is shown in Table 15.2.

With MU-TERM 5.0, we solve 8 examples more than with the previous version and, moreover, 2 examples more than AProVE. Therefore, nowadays, MU-TERM seems to be the most powerful tool for being used as backend for MTT.

The application, its source code, and all the related information concerning MTT are available in

http://www.lcc.uma.es/~duran/MTT

Therefore, any improvement on these termination tools (in our specific case, in MU-TERM) can be useful for MTT and, consequently, for proving termination of Maude programs. In the near future, MTT will be adapted to take advantage of the new features implemented in MU-TERM when proofs of termination of Maude programs are attempted [Dur11].

16

Conclusions

Proving termination of programs has been a hard and fashionable task in recent years. In this thesis, we have addressed the challenge of proving termination of Maude programs.

Although several papers have previously studied how to deal with termination of a subset of Maude programs, all of them followed a transformational approach and the practical results were not really satisfactory. Therefore, there was an important need to develop direct methods for proving termination of Maude programs. In the last few years, we have worked to extend the DP framework to verify a number of termination properties of variants of TRSs, most of which are able to represent inherent features of Maude programs.

With regard to the default computational mechanism, the Maude strategy is call by value and admits syntactic annotations used as *replacement restrictions* to become 'more lazy', thus (hopefully) avoiding nontermination. Following this, we have developed Part I of this thesis, where we have dealt with proving termination of innermost context-sensitive rewriting that covers these features of Maude programs. We have investigated the structure of infinite innermost context-sensitive rewrite sequences since this analysis is essential to provide an appropriate definition of innermost context-sensitive dependency pair, and the related notions of innermost μ-chains, etc. that have allowed us to define our ICSDP framework. Techniques for proving termination of innermost *CSR* were first investigated following a transformational approach. Thus, our framework is the first direct method for proving termination of ICSR. We have defined several specific processors and implemented our results in the termination tool MU-TERM showing that our framework dramatically improves the performance of previous transformational methods.

With respect to specific features of Maude programs, we have addressed the declaration of equational attributes, specifically, associative and/or commutative theories. Computationally, this corresponds to rewriting modulo such equations. Even though there were several works and even implementations

for proving termination of several combinations of equational theories, we have provided an appropriate notion of minimal term and of chain for $A \vee C$-theories that has been used to develop an $A \vee C$-dependency pair framework for proving termination of $A \vee C$-rewrite theories for the first time. After implementing the techniques described in Part II, MU-TERM has become a powerful tool for proving termination of $A \vee C$-rewrite theories.

Therefore, although Maude incorporates many other features that must be taken into account to fully capture termination of its programs, this thesis constitutes a great step forward in dealing with some of them in a very successful way.

With regard to future work, we think interesting to address the following issues and problems:

- Improve the estimation of the $A \vee C$ graph in the presence of associative axioms.
- Investigate the use of argument filterings [KNT99] in the processor of reduction pairs with usable rules and equations.
- Investigate proofs of innermost non-μ-termination and $A \vee C$-nontermination.
- Integrate the results about conditional termination of [SG10] in our framework for proving termination of Maude programs.

Obviously, we also plan to integrate Part I and Part II, i.e., investigate the (innermost) termination of $A \vee C$-rewrite theories having replacement restrictions since, frequently, syntactic annotations and attributes come together in Maude programs. This is the first main goal and after that, we would like to extend it to order-sorted rewriting following [LM08] and to investigate the integration with conditions and memberships. There is no doubt that these objectives are very ambitious and if it is possible, we will try to follow the same methodology: direct methods inside the DP framework.

Conclusiones

Probar terminación de programas es una tarea dura y muy de moda en los últimos años. En esta tesis, nos hemos centrado en probar terminación de programas Maude.

Aunque varios trabajos han estudiado con anterioridad como manejar termiación de un subconjunto de programas Maude, todos ellos seguían una aproximación transformacional y los resultados prácticos no habían sido realmente satisfactorios. Por lo tanto, existía una necesidad importante de desarrollar métodos directos para probar terminación de programas Maude. En los últimos años, hemos trabajado extendiendo el marco de pares de dependencia para verificar varias propiedades de terminación de variantes de sistemas de reescritura de términos, la mayoría capaz de representar características inherentes de los programas Maude.

Con respecto al mecanismo computacional por defecto, la estrategia de Maude es de llamada por valor y admite anotaciones sintácticas usadas como restricciones de reemplazamiento para convertirlo en más perezoso evitando así (con suerte) la no terminación. Siguiendo esto, hemos desarrollado la Parte I de esta tesis, donde hemos llevado a cabo la verificación de la terminación de la reescritura sensible al contexto innermost (RSCI), que cubre estas características de los programas Maude. Hemos investigado la estructura de las secuencias infinitas de reescritura sensible al contexto innermost ya que este análisis es esencial para llegar a una definición apropiada de par de dependencia sensible al contexto innermost y las correspondientes nociones de μ-cadena innermost, etc. que nos han permitido definir nuestro marco. Técnicas para probar terminación de la reescritura sensible al contexto innermost han sido investigadas con anterioridad siguiendo una aproximación transformacional. Por lo tanto, nuestro marco es el primer método directo para probar terminación de la RSCI. Hemos definido varios procesadores específicos e implementado nuestros resultados en la herramienta de termiación MU-TERM demostrando que nuestro marco, mejora dramáticamente el rendimiento de los métodos transformacionales anteriores.

Con respecto a características específicas de programas Maude, nos hemos centrado en la declaración de atributos ecuacionales, en concreto, teorías asociativas y/o conmutativas. Computacionalmente, se corresponde con reescritura módulo esas ecuaciones. A pesar de que existían varios trabajos e incluso

implementaciones para probar terminación de varias combianciones de teorías ecuacionales, hemos proporcionado, por primera vez, una noción apropiada de término minimal y cadena para teorías $A \vee C$ que puede ser usada para desarrollar un marco de pares de dependencia para probar terminación de teorías de reescritura $A \vee C$. Después de implementar las técnicas descritas en la Parte II, MU-TERM se ha convertido en una potente herramienta para probar terminación de teorías de reescritura $A \vee C$.

Por lo tanto, aunque Maude incorpora muchas otras características que tendrían que deben ser tenidas en cuenta para capturar completamente la terminación de sus programas, esta tesis constituye un gran paso adelante para tratar con algunas de ellas de un modo muy satisfactorio.

En relación con el trabajo futuro, creemos que sería interesante investigar los siguientes temas:

- Mejorar la estimación del grafo $A \vee C$ en presencia de axiomas asociativos.
- Investigar el uso del filtrado de argumentos [KNT99] en el procesador de pares de reducción con reglas y ecuaciones usables.
- Investigar pruebas de no μ-terminación innermost y no terminación $A \vee C$.
- Integrar los resultados sobre terminación condicional de [SG10] en nuestro marco para probar terminación de Maude.

Obviamente, también planeamos integrar Parte I y II, es decir, investigar la terminación innermost de teorías $A \vee C$ con restricciones de reemplazamiento ya que, frecuentemente, las anotaciones sintácticas y los atributos aparecen juntos en los programas Maude. Este es el primer objetivo y principal y, despues de ello, nos gustaría extender el marco a la reescritura con tipos siguiendo [LM08] e investigar su integración con condiciones y merberships. No hay duda que estos objetivos son muy ambiciosos y, si es posible, intentaremos seguir la misma metodología que hasta ahora: métodos directos dentro del marco de los pares de dependencia.

Conclusions

Provar terminació de programes és una tasca dura i molt de moda en els últims anys. En aquesta tesi, ens hem centrat a provar terminació de programes Maude.

Encara que diversos treballs han estudiat amb anterioritat com abordar la terminació d'un subconjunt de programes Maude, tots ells seguien una aproximació transformacional i els resultats pràctics són realment satisfactoris. Per tant, existeix una necessitat important de desenvolupar mètodes directes per a provar terminació de programes Maude. En els últims anys, hem treballat intentant estendre el marc de parells de dependència per a verificar diverses propietats de terminació de variants de sistemes de reescriptura de termes, la majoria capaç de representar característiques inherents dels programes Maude.

Pel que fa al mecanisme computacional per defecte, l'estratègia de Maude és de crida per valor i admet anotacions sintàctiques usades com restriccions de reemplaament per a convertir-lo en mes peresós evitant així (amb sort) la no terminació. Seguint açó, hem desenvolupat la Part I d'aquesta tesi, on hem portat a terme la verificació de la terminació de la reescriptura sensible al context innermost (RSCI), que cobreix aquestes característiques dels programes Maude. Hem investigat l'estructura de les seqüències infinites de reescriptura sensible al context innermost ja que aquesta anàlisi és essencial per arribar a una definició apropiada de parell de dependència sensible al context innermost i les corresponents nocions de μ-cadena innermost, etc. que ens han permès definir el nostre marc. Tècniques per a provar terminació de la reescriptura sensible al context innermost han estat investigades amb anterioritat seguint una aproximació transformacional. Per tant, el nostre marc és el primer mètode directe per a provar terminació de la RSCI. Hem definit diversos processadors específics i implementat els nostres resultats en l'eina de terminació MU-TERM demostrant que el nostre marc millora dramàticament el rendiment dels mètodes transformacionals anteriors.

Pel que fa a característiques específiques de programes Maude, ens hem centrat en la declaració d'atributs equacionals, en concret, teories associatives i/o commutatives. Computacionalment, es correspon amb reescriptura mòdul les equacions. A pesar que existien diversos treballs i fins i tot implementacions per a provar terminació de diverses combinacions de teories de reescriptura, hem proporcionat, per primera vegada, una noció apropiada de terme minimal

i cadena per a teories $A \vee C$ que pot ser usada per a desenvolupar un marc de parells de dependència per a provar terminació de teories de reescriptura $A \vee C$. Després d'implementar les tècniques descrites en la Part II, MU-TERM s'ha convertit en una potent eina per a provar terminació de teories de reescriptura $A \vee C$.

Per tant, encara que Maude incorpora moltes altres característiques que tindrien que ser tingudes en compte per a capturar completament la terminació dels seus programes, aquesta tesi constitueix un gran pas avant per a tractar amb algunes d'elles d'una manera molt satisfactòria.

En relació amb el treball futur, creiem que seria interessant investigar els temes següents:

- Millorar l'estimació del grafo $A \vee C$ en presència d'axiomes associatius.
- Investigar l'ús del filtrat d'arguments [KNT99] en el processador de parells de reducción amb regles i equacions usables.
- Investigar proves de no μ-terminació innermost i no terminació $A \vee C$.
- Integrar els resultats sobre terminació condicional de [SG10] en el nostre marc per a provar terminació de Maude.

Òbviament, també planegem integrar Part I i II, és a dir, investigar la terminació innermost de teories $A \vee C$ amb restriccions de reemplaament ja que, sovint, les anotacions sintàctiques i els atributs apareixen junts en els programes Maude. Este és el primer objectiu i principal i, després d'això, ens agradaria estendre el marc a la reescriptura amb tipus seguint [LM08] i investigar la seua integració amb condicions i merberships. No hi ha dubte que estos objectius són molt ambiciosos i, si és possible, intentarem seguir la mateixa metodologia que fins ara: mètodes directes dins del marc dels parells de dependència.

Bibliography

[AAC+08] E. Albert, P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini. Termination Analysis of Java Bytecode. In G. Barthe and Frank S. Boer, editors, *Proc. of the 10th IFIP WG 6.1 international conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'08*, pages 2–18, Springer-Verlag, 2008.

[AEF+08] B. Alarcón, F.Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving context-sensitive dependency pairs. In I. Cervesato, H. Veith and A. Voronkov, editors, *Proc. of 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of LNAI, pages 636–651. Springer-Verlag, 2008.

[AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

[AGIL07] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.

[AGL06] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of LNCS, pages 297–308. Springer-Verlag, 2006.

[AGL07] B. Alarcón, R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

[AGL10] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. *Information and Computation*, 208:922–968, 2010.

[AGLM11] B. Alarcón, R. Gutiérrez, S. Lucas, and J. Meseguer. A Dependency Pair Framework for $A \vee C$-Termination. Technical Report, http://hdl.handle.net/10251/10797, DSIC-ELP, UPV, 2011.

[AGLN10] B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. Proving Termination Properties with MU-TERM. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology and Software Technology, AMAST'10*, volume 6486 of LNCS, pages 201–208. Springer-Verlag, 2010.

[AL07] B. Alarcón and S. Lucas. Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, volume 4720 of LNAI, pages 73–87. Springer-Verlag, 2007.

[AL09] B. Alarcón, S. Lucas. Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting. *Electronic Notes in Theoretical Computer Science*, 248:3–17, 2009.

[AL11] B. Alarcón, S. Lucas. Innermost Termination of Context-Sensitive Rewriting. Technical Report, http://hdl.handle.net/10251/10796, DSIC-ELP, UPV, 2011.

[ALM10] B. Alarcón, S. Lucas, J. Meseguer, A Dependency Pair Framework for $A \vee C$-Termination. In P. Ölveczky, editor, *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of LNCS, pages 35–51. Springer-Verlag, 2010.

[AM93] G. Aguzzi and U. Modigliani. Proving termination of logic programs by transforming them into equivalent term rewriting systems. In R. Shyamasundar, editor, *Proc. of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS93*, volume 761 of LNCS, pages 114–124. Springer-Verlag, 1993.

[BFR00] C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. In D. McAllester, editor, *Proc of the 17th International Conference on Automated Deduction,*

CADE'00, volume 1831 of LNAI, pages 346–364. Springer-Verlag, 2000.

[BM06] R. Bruni and J. Meseguer. Semantic foundations for generalized rewrite theories. *Theoretical Computer Science* 351(1):386–414, 2006.

[BN98] F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.

[Bor03] C. Borralleras. Ordering-based methods for proving termination automatically. PhD Thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, 2003.

[BJM00] A. Bouhoula, J.-P. Jouannaud and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science* 236:35–132, 2000.

[BM03] R. Bruni, J. Meseguer. Generalized rewrite theories. In J.C.M. Baeten, J.K. Lenstra, J. Parrow and G.J. Woeginger, editors, *Proc of the 30th international conference on automata, languages and programming, ICALP'03*, volume 2719 of LNCS, pages 252–266. Springer-Verlag, 2003.

[BMS05] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination of polynomial programs. In *Proc. of the 6th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 05*, volume 3385 of LNCS, pages 113–129. Springer-Verlag, 2005.

[CDEL+07] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. volume 4350 of LNCS. Springer-Verlag, 2007.

[CLS05] M. Codish, V. Lagoon, and P. Stuckey. Testing for termination with monotonicity constraints. In M. Gabbrielli, and G. Gupta, editors, *Proc. of the 21th International Conference on Logic Programming, ICLP 05*, volume 3668 of LNCS, pages 326–340. Springer-Verlag, 2005.

[CMMU03] E. Contejean, C. Marché, B. Monate, X. Urbain, Proving Termination of Rewriting with C*i*ME. In A. Rubio, editor, *Proc. of the 6th International Workshop on Termination, WST'03*, pages 71–73. 2003.

[CPR06] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. In *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI'06*, pages 415–426. ACM Press, 2006.

[CS02] M. Colon and H. B. Sipma. Practical methods for proving program termination. In E. Brinksma and K. Larsen, editors, *Proc. of the 14th International Conference on Computer Aided Verification, CAV 02*, volume 2034 of LNCS, pages 442–454. Springer-Verlag, 2002.

[CT99] M. Codish and C. Taboch. A semantic basis for the termination analysis of logic programs. *The Journal of Logic Programming*, 41(1):103–123, 1999.

[Dau92] M. Dauchet. Simulation of Turing machines by a regular rewrite rule. *Theoretical Computer Science*, 103(2):409–20, 1992.

[DD94] D. De Schreye and S. Decorte. Termination of logic programs: The neverending story. *Journal of Logic Programming*, 19/20:199–260, 1994.

[Der79] N. Dershowitz. A note on simplification orderings. *Information Processing Letters*, 9(5):212–215, 1979.

[Der82] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[Der87] N. Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation* 3:69–116, 1987.

[Der04] N. Dershowitz. Termination by Abstraction. In B. Demoen and V. Lifschitz, editors, *Proc. of 20th International Conference on Logic Programming, ICLP'04*, volume 3132 of LNCS, pages 1–18. Springer-Verlag, 2004.

[DLSS01] N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic

programs. *Applicable Algebra in Engineering, Communication and Computing*, 12(1/2):117-156, 2001.

[DLM+04] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Termination of Membership Equational Programs. In *Proc. of 2004 ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147–158. ACM Press, 2004.

[DLM08a] F. Durán, S. Lucas, and J. Meseguer. MTT: The Maude Termination Tool (System Description). In A. Armando, P. Baumgartner and G. Dowek, editors, *Proc. of 4th International Joint Conference on Automated Reasoning, IJCAR'08*, volume 5195 of LNAI, pages 313–319. Springer-Verlag, 2008.

[DLM08b] F. Durán, S. Lucas, and J. Meseguer. Operational Termination in Rewriting Logic. Technical Report 2008. http://www.dsic.upv.es/ slucas/tr08.pdf

[DLM09a] F. Durán, S. Lucas, and J. Meseguer. Methods for proving termination of rewriting-based programming languages. *Electronic Notes in Theoretical Computer Science*, 248:93–113, 2009.

[DLM09b] F. Durán, S. Lucas, J. Meseguer. Termination Modulo Combinations of Equational Theories, In S. Ghilardi, R. Sebastiani, editors, *Proc of 7th International Symposium on Frontiers of Combining Systems, FroCoS'09*, volume 5749 of LNAI, pages 246–262. Springer-Verlag, 2009.

[DLM+08] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, 21(1-2):59–88, 2008.

[DS02] D. De Schreye and A. Serebrenik. Acceptability with general orderings. In A. Kakas and F. Sadri, editors, *Computational Logic: Logic Programming and Beyond*, volume 2407 of LNCS, pages 187–210. Springer-Verlag, 2002.

[Dur11] F. Durán. *Personal communication*, January 2011.

[Fal04] S.P. Falke. Automated Termination Analysis for Equational Rewriting, Diplomarbeit, Fakultät für Informatik, Rheinisch-Westfälische Technische Hochschule Aachen, 2004.

[Fer05] M. L. Fernández. Relaxing monotonicity for innermost termination. *Information Processing Letters*, 93(3):117–123, 2005.

[FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 52–66, ACM Press, 1985.

[FN97] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.

[FR99] M.C.F. Ferreira and A.L. Ribeiro. Context-Sensitive AC-Rewriting. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, volume 1631 of LNCS, pages 286–300. Springer-Verlag, 1999.

[GAO02] J. Giesl, T. Arts, and E. Ohlebusch. Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.

[GHW03] A. Geser, D. Hofbauer, and J. Waldmann. Match-bounded string rewriting systems. *Applicable Algebra in Engineering, Communication and Computing*, 15(3): 149–171, 2004.

[GHWZ07] A. Geser, D. Hofbauer, J. Waldmann, and H. Zantema. On tree automata that certify termination of left-linear term rewriting systems. *Information and Computation*, 205(4):512–534, 2007.

[Gie95] J. Giesl. Termination analysis for functional programs using term orderings. In A. Mycroft, editor, *Proc. of the 2nd International Static Analysis Symposium, SAS'95*, volume 983 of LNCS, pages 154–171, Springer-Verlag, 1995.

[GK01] J. Giesl, D. Kapur. Dependency Pairs for Equational Rewriting. In A. Middeldorp, editor, *Proc. of 12th International Conference of Rewriting Techniques and Applications, RTA'01*, volume 2051 of LNCS, pages 93–108. Springer-Verlag, 2001.

[GL02a] B. Gramlich and S. Lucas. Simple Termination of Context-Sensitive Rewriting. In B. Fischer and E. Visser, editors, *Proc. of III ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02*, pages 29–41. ACM Press, 2002.

[GL02b] B. Gramlich and S. Lucas. Modular Termination of Context-Sensitive Rewriting. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'02*, pages 50–61. ACM Press, 2002.

[GLU08] R. Gutiérrez, S. Lucas, and X. Urbain. Usable Rules for Context-Sensitive Rewrite Systems. In A. Voronkov, editor, *Proc. of the 19th International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of LNCS, pages 126–141. Springer-Verlag, 2008.

[GL10] R. Gutiérrez and S. Lucas. Proving Termination in the Context-Sensitive Dependency Pair Framework. In P. Ölveczky, editor, *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10* volume 6381 of LNCS, pages 19–35. Springer-Verlag, 2010.

[GM92] J. Goguen, J. Meseguer. Order-sorted algebra I: Equational deduction for multiple inheritance, overloading, exceptions and partial operations. *Theoretical Computer Science*, 105:217–273, 1992.

[GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of X International Conference on Rewriting Techniques and Applications, RTA'99*, volume 1631 of LNCS, pages 271–285, Springer-Verlag, 1999.

[GM02a] J. Giesl and A. Middeldorp. Innermost termination of context-sensitive rewriting. In M. Ito and M. Toyama, editors, *Proc. of 6th International Conference on Developments in Language Theory, DLT'02*, volume 2450 of LNCS, pages 231–244. Springer-Verlag, 2003.

[GM02b] J. Giesl and A. Middeldorp. Innermost termination of context-sensitive rewriting. Aachener Informatik-Berichte (AIB-2002-04), RWTH Aachen, ISSN 0935-3232, 2002.

[GM04] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.

[Gra95] B. Gramlich. Abstract Relations between Restricted Termination and Confluence Properties of Rewrite Systems. *Fundamenta Informaticae*, 24:3–23, 1995.

[Gra96] B. Gramlich. On Proving Termination by Innermost Termination. In H. Ganzinger, editor, *Proc. 7th Int. Conf. on Rewriting Techniques and Applications, RTA'96*, volume 1103 of LNCS, pages 93–107. Springer-Verlag, 1996.

[GRS+10] J. Giesl, M. Raffelsieper, P. Schneider-Kamp, S. Swiderski, and R. Thiemann Automated Termination Proofs for Haskell by Term Rewriting. ACM Transactions on Programming Languages and Systems, 33(2), 7:1–7:39, 2010.

[GST06] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, volume 4130 of LNAI, pages 281–286. Springer-Verlag, 2006. Available at http://www-i2.informatik.rwth-aachen.de/AProVE.

[GTS04] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors, *Proc. of XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04*,volume 3452 of LNAI, pages 301–331. Springer-Verlag, 2004.

[GTS05] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and Disproving Termination of Higher-Order Functions. In B. Gramlich, editor, *Proc. of 5th International Workshop on Frontiers of Combining Systems, FroCoS'05*, volume 3717:of LNAI, pages 216–231. Springer-Verlag, 2005.

[GTSF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006.

[Gut10] R. Gutiérrez. Automatic Proofs of Termination of Context-Sensitive Rewriting PhD. Thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain, 2010.

[GWM+00] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.

[HL78] G. Huet, D.S. Lankford. On the uniform halting problem for term rewriting systems. Rapport Laboria 283. Institut National de Reserche en Informatique et en Automatique, Le Chesnay, France, 1978.

[HM04] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In V. van Oostrom, editor, *Proc. of XV International Conference on Rewriting Techniques and Applications, RTA'04*, volume 3091 of LNCS, pages 249–268. Springer-Verlag, 2004.

[HM05] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. *Information and Computation*, 199:172–199, 2005.

[HM07] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474–511, 2007.

[HPW92] P. Hudak, S.J. Peyton-Jones, and P. Wadler. Report on the Functional Programming Language Haskell: a non–strict, purely functional language. *Sigplan Notices*, 27(5):1–164, 1992.

[KB70] D. E. Knuth and P.B. Bendix. Simple word problems in universal algebra. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[KL80] S. Kamin and J.-J. Levy. Attempts for generalizing the recursive path ordering. Unpublished note, Dept. of Computer Science, University of Illinois, 1980.

[KM09] M. Korp and A. Middeldorp. Match-Bounds Revisited. *Information and Computation* 207(11):1259–1283, 2009.

[KKS98] M. R. K. Krishna Rao, D. Kapur, and R. Shyamasundar. Transformational methodology for proving termination of logic programs. *Journal of Logic Programming*, 34(1):1–42, 1998.

[KNT99] K. Kusakari, M. Nakamura, and Y. Toyama. Argument Filtering Transformation. In G. Nadathur, editor, *Proc. of International Conference on Principles and Practice of Declarative Programming, PPDP'99*, volume 1702 of LNCS, pages 47–61. ACM Press, 1999.

[Kus00] K. Kusakari. Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems. PhD. Thesis, School of Information Science, JAIST, 2000.

[KMU02] K. Kusakari, C. Marché, and X. Urbain. Termination of associative-commutative rewriting using dependency pairs criteria. Research Report 1304, LRI, 2002.

[KNT06] K. Kusakari, M. Nakamura, Y. Toyama. Elimination Transformations for Associative-Commutative Rewriting Systems. *Journal of Automated Reasoning* 37:205–229, 2006.

[KT01] K. Kusakari, Y. Toyama. On Proving AC-Termination by AC-Dependency Pairs. *IEICE Transactions on Information and Systems*, E84-D, 604–612, 2001.

[Lan75] D. S. Lankford. Canonical algebraic simplification in computational logic. Technical Report ATP-25, Department of Mathematics, University of Texas, Austin, 1975.

[Lan77] D.S. Lankford. Some approaches to equality for computational logic: A survey and assessment. Memo ATP-36. Automatic Theorem Proving Project. University of Texas, 1977.

[Lan79] D.S. Lankford. On proving term rewriting systems are Noetherian. Memo MTP-3. Louisiana Tech. University, 1979.

[LJB01] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. *ACM Symposium on Principles of Programming Languages, POPL'01*, pages 81–92. ACM Press, 2001.

[LM08] S. Lucas and J. Meseguer. Order-Sorted Dependency Pairs. In S. Antoy, editor, *Proc. of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08*, pages 108–119. ACM Press, 2008.

[LM09] S. Lucas and J. Meseguer. Operational Termination of Membership Equational Programs: the Order-Sorted Way. In G. Rosu, editor, *Proc. of the 7th International Workshop on Rewriting Logic and its Applications, WRLA'08*, *Electronic Notes in Theoretical Computer Science*, 238(3):207–225, 2009.

[LMS03] V. Lagoon, F. Mesnard, and P. J. Stuckey. Termination analysis with types is more accurate. In *Proc. of the 19th International Conference on Logic Programming, ICLP'03*, volume 2916 of LNCS, pages 254–268. Springer-Verlag, 2003.

[Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, volume 1099 of LNCS, pages 122–133. Springer-Verlag, 1996.

[Luc98] S. Lucas. Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, 1998.

[Luc01a] S. Lucas. Termination of Rewriting With Strategy Annotations. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, volume 2250 of LNAI, pages 669–684. Springer-Verlag, 2001.

[Luc01b] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 82–93, ACM Press, 2001.

[Luc01c] S. Lucas. *Personal communication* to authors of [GM02a], 2001-2002.

[Luc02] S. Lucas. Context-Sensitive Rewriting Strategies. *Information and Computation*, 178(1):293–343, 2002.

[Luc04] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of XV International Conference on Rewriting Techniques and Applications, RTA'04*, volume 3091 of LNCS, pages 200–209. Springer-Verlag, 2004. Available at http://zenon.dsic.upv.es/muterm.

[Luc06] S. Lucas. Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation*, 204(12):1782–1846, 2006.

[Mar94] M. Marchiori. Logic programs as term rewriting systems. In G. Levi, and M. Rodrguez-Artalejo, editors, *Proc. of the 4th International Conference on Algebraic and Logic Programming, ALP'94*, volume 850 of LNCS, pages 223–241. Springer-Verlag, 1994.

[Mar96] M. Marchiori. Proving existential termination of normal logic programs. In M. Wirsing and M. Nivat, editors, *Proc. of the 5th International Conference on Algebraic Methodology and Software Technology, AMAST'96*, volume 1101 of LNCS, pages 375–390. Springer-Verlag, 1996.

[McC60] J. McCarthy. Recursive Functions of Symbolic Expressions and their Computations by Machine, Part I. *Communications of the ACM*, 3(4):184–195, 1960.

[Mes92] J. Meseguer. Conditional rewriting logic as a model of concurrency. *Theoretical Computer Science*, 96:73–155, 1992.

[Mes98] J. Meseguer. Membership algebra as a logical framework for equational specification. In F. Parisi-Presicce, editor, *Recent Trends in Algebraic Development Techniques*. volume 1376 of LNCS, pages 18–61. Springer-Verlag, 1998.

[Mid01] A. Middeldorp. Approximating dependency graphs using tree automata techniques. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the International Joint Conference on Automated Reasoning, IJCAR'01*, volume 2083 of LNAI, pages 593–610. Springer-Verlag, 2001.

[Mid02] A. Middeldorp, Approximations for strategies and termination. *Electronic Notes in Computer Science*, 70(6), 2002.

[MN70] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proc. Third Hawaii Int. Conf. System Science*, pages 789–792, 1970.

[MU98] C. Marché, X. Urbain. Termination of Associative-Commutative Rewriting by Dependency Pairs, In T. Nipkow, editor, *Proc of 9th International Conference on Rewriting Techniques and Applications, RTA'98*, volume 1379 of LNCS, pages 241–255. Springer-Verlag, 1998.

[MU04] C. Marché, X. Urbain, Modular and Incremental Proofs of AC-Termination, *Journal of Symbolic Computation* 38:873–897, 2004.

[MZ07] C. Marché and H. Zantema. The Termination Competition. In F. Baader, editor, *Proc of 18th International Conference on Rewriting Techniques and Applications, RTA'07*, volume 4533 of LNCS, pages 303–313. Springer-Verlag, 2007.

[NSEP92] E.G.J.M.H. Nöcker, J.E.W. Smetsers, M.C.J.D. van Eekelen, and M.J. Plasmeijer. Concurrent Clean. In E.H.L. Aarts, J. Leeuwen, and M. Rem, editors, *Proc. of Parallel Architectures and Languages Europe, PARLE'91*, volume 506 of LNCS, pages 202–219. Springer-Verlag, 1992.

[OBEG10] C. Otto, M. Brockschmidt, C. von Essen and J. Giesl. Automated Termination Analysis of Java Bytecode by Term Rewriting. *Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA'10*, pages 259–276, 2010.

[Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting.* Springer-Verlag, 2002.

[ÖL96] P. C. Ölveczky and O. Lysne. Order-Sorted Termination: The Unsorted Way. In M. Hanus, and M. Rodrguez-Artalejo, editors, *Proc of Algebraic and Logic Programming, ALP'96*, volume 1139 of LNCS, pages 92–106, Springer-Verlag, 1996.

[PS81] G.E. Peterson, M.E. Stickel. Complete Sets of Reductions for Some Equational Theories, *Journal of the ACM*, 28(2) 233–264, 1981.

[Rub02] A. Rubio. A Fully Syntactic AC-RPO. *Information and Computation*, 178(2): 515–533, 2002.

[SG09] F. Schernhammer and B. Gramlich. VMTL - A Modular Termination Laboratory. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*, volume 5595 of LNCS, pages 285–294. Springer-Verlag, 2009.

[SG10] F. Schernhammer and B. Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. *Journal of Logic and Algebraic Programming*, 79(7): 659–688, 2010.

[SGN09] P. Schneider-Kamp, J. Giesl and M. T. Nguyen. The Dependency Triple Framework for Termination of Logic Programs. In D. De Schreye, editor, *Proc. of the 19th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR'09*, volume 6037 of LNCS, pages 37–51. Springer-Verlag, 2009.

[SGST06] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated termination analysis for logic programs by term rewriting. In G. Puebla, editor, *Proc. of the 16th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR'06*, volume 4407 of LNCS, pages 177–193. Springer-Verlag, 2006.

[Sma04] J.-G. Smaus. Termination of logic programs using various dynamic selection rules. In B. Demoen and V. Lifschitz, editors, *Proc. of the 20th International Conference on Logic Programming, ICLP'04*, volume 3132 of LNCS, pages 43–57. Springer-Verlag, 2004.

[Ste06] C. Stein. Das Dependency Pair Framework zur automatischen Terminierungsanalyse von Termersetzung modulo Gleichungen. Diplomarbeit, Fakultät für Informatik, Rheinisch-Westfälische Technische Hochschule Aachen, 2006.

[TeR03] TeReSe. *Term Rewriting Systems*. Cambridge University Press, 2003.

[TGS04] R. Thiemann, J. Giesl, P. Schneider-Kamp. Improved Modular Termination Proofs Using Dependency Pairs. In D. Basin and M. Rusinowitch, editors, *Proc. of 2nd International Joint*

Conference on Automated Reasoning, IJCAR'04, volume 3097, of LNAI, pages 75–90. Springer-Verlag, 2004.

[Thi07] R. Thiemann. The DP Framework for Proving Termination of Term Rewriting. PhD Thesis. Available as Technical Report AIB-2007-17, RWTH Aachen University, Germany, 2007.

[Tiw04] A. Tiwari. Termination of linear programs. In R. Alur and D. A. Peled, editors, *Proc. of the 16th International Conference on Computer Aided Verification, CAV'04*, volume 3114 of LNCS, pages 387–390. Springer-Verlag, 2004.

[Urb04] X. Urbain, Modular & Incremental Automated Termination Proofs, *Journal of Automated Reasoning*, 32:315–355, 2004.

[Wal09] J. Waldmann. Report on the Termination Competition 2008. *Proc. of 10th International Workshop on Termination, WST'09*, pages 100–111, 2009.

[Xi02] H. Xi. Dependent types for program termination verification. *Higher-Order and Symbolic Computation*, 15(1):91–131, 2002.

[Zan95] H. Zantema. Termination of term rewriting by semantic labelling. Fundamenta Informaticae, 24:89–105, 1995.

[Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of VII International Conference on Rewriting Techniques and Applications, RTA'97*, volume 1232 of LNCS, pages 172–186. Springer-Verlag, 1997.

Index

Part III

Publications Associated to the Thesis

17

List of Publications

1. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 2006.

2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph**. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with** MU-TERM. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.

4. B. Alarcón, and S. Lucas. **Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs**. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 73–87, Springer-Verlag, 2007.

5. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. **Improving Context-Sensitive Dependency Pairs**. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of XV International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *Lecture Notes in Computer Science*, pages 636–651. Springer-Verlag, 2008.

6. B. Alarcón, and S. Lucas. **Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting**. *Electronic Notes in Theoretical Computer Science*, 248:3-17, 2009. Selected papers from the 8th Spanish Conference on Programming and Languages, PROLE'08.

7. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. *Information and Computation*, 208:922–968, 2010.

8. B. Alarcón, S. Lucas, and J. Meseguer. **A Dependency Pair Framework for $A\vee C$-Termination**. In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 35–51. Springer-Verlag, 2010.

9. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. **Proving Termination Properties with** MU-TERM. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology And Software Technology, AMAST'10*, volume 6486 of *Lecture Notes in Computer Science*, pages 201–208. Springer-Verlag, 2010.

10. B. Alarcón, and S. Lucas. **Innermost Termination of Context-Sensitive Rewriting**. Technical Report, http://hdl.handle.net/10251/10796, DSIC-ELP, UPV, 2011.

11. B. Alarcón, R. Gutiérrez, S. Lucas, and J. Meseguer. **A Dependency Pair Framework for $A\vee C$-Termination**. Technical Report, http://hdl.handle.net/10251/10797, DSIC-ELP, UPV, 2011.

18

Publications (full text)

18.1 Context-Sensitive Dependency Pairs

1. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 2006.

Context-Sensitive Dependency Pairs*

Beatriz Alarcón, Raúl Gutiérrez, and Salvador Lucas

DSIC, Universidad Politécnica de Valencia, Spain
{balarcon , rgutierrez , slucas}@dsic.upv.es

Abstract. Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. Although there is a good number of techniques for proving termination of context-sensitive rewriting (*CSR*), the dependency pair approach, one of the most powerful techniques for proving termination of rewriting, has not been investigated in connection with proofs of termination of *CSR*. In this paper, we show how to use dependency pairs in proofs of termination of *CSR*. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of *CSR*.

Keywords: Dependency pairs, term rewriting, program analysis, termination.

1 Introduction

A *replacement map* is a mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f of a signature $\mathcal{F}$ [Luc98]. We use them to discriminate the argument positions on which the rewriting steps are allowed. In this way, for a given Term Rewriting System (TRS [Ohl02, Ter03]), we obtain a restriction of rewriting which we call *context-sensitive rewriting* (*CSR* [Luc98, Luc02]). In *CSR* we only rewrite μ-replacing subterms: t_i is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$; every term t (as a whole) is μ-replacing by definition. With *CSR* we can *achieve* a terminating behavior with non-terminating TRSs, by pruning (all) infinite rewrite sequences. Proving termination of *CSR* has been recently recognized as an interesting problem with several applications in the fields of term rewriting and programming languages (see [DLMMU06, GM04, Luc02, Luc06]).

Several methods have been developed for proving termination of *CSR* under a replacement map μ for a given TRS $\mathcal{R}$ (i.e., for proving the *μ-termination* of $\mathcal{R}$). In particular, a number of transformations which permit to treat termination of *CSR* as a standard termination problem have been described (see

* This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grant TIN 2004-7943-C04-02, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

S. Arun-Kumar and N. Garg (Eds.): FSTTCS 2006, LNCS 4337, pp. 298–309, 2006.
© Springer-Verlag Berlin Heidelberg 2006

[GM04, Luc06] for recent surveys). Direct techniques like polynomial orderings and the context-sensitive version of the recursive path ordering have also been investigated [BLR02, GL02, Luc04b, Luc05]. Up to now, however, the *dependency pairs method* [AG00, GAO02, GTS04, HM04], one of the most powerful techniques for proving termination of rewriting, has not been investigated in connection with proofs of termination of *CSR*. In this paper, we address this problem.

Roughly speaking, given a TRS $\mathcal{R}$, the dependency pairs associated to $\mathcal{R}$ conform a new TRS $\mathsf{DP}(\mathcal{R})$ which (together with $\mathcal{R}$) determines the so-called *dependency chains* whose finiteness or infiniteness characterize termination of $\mathcal{R}$. Given a rewrite rule $l \to r$, we get dependency pairs $l^\sharp \to s^\sharp$ for all subterms s of r which are rooted by a defined symbol[1]; the notation $t^\sharp$ for a given term t means that the root symbol f of t is *marked* thus becoming $f^\sharp$ (often just capitalized: F). A chain of dependency pairs is a sequence $u_i \to v_i$ of dependency pairs such that $\sigma(v_i)$ rewrites to $\sigma(u_{i+1})$ for some substitution σ and $i \geq 1$. The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. These basic intuitions are valid for *CSR*, although some important differences arise.

Example 1. Consider the following TRS $\mathcal{R}$ [GM99, Example 1]:

```
c -> a      f(a,b,X) -> f(X,X,X)
c -> b
```

together with $\mu(\mathtt{f}) = \{3\}$. As shown by Giesl and Middeldorp, among all existing transformations for proving termination of *CSR*, only the *complete* Giesl and Middeldorp's transformation [GM04] (yielding a TRS $\mathcal{R}_C^\mu$) could be used in this case, but no concrete proof of termination for $\mathcal{R}_C^\mu$ is known yet. Furthermore, $\mathcal{R}_C^\mu$ has 13 dependency pairs and the dependency graph contains many cycles. In contrast, $\mathcal{R}$ has only *one* context-sensitive (CS-)dependency pair

```
F(a,b,X) -> F(X,X,X)
```

and the corresponding dependency graph has *no* cycle (due to the replacement restrictions, since we extend μ by $\mu(\mathtt{F}) = \{3\}$). As we show below, a direct (and automatic) proof of μ-termination of $\mathcal{R}$ is easy now.

Basically, the subterms in the right-hand sides of the rules which are considered to build the CS-dependency pairs must be *μ-replacing* terms. However, this is not sufficient to obtain a correct approximation. The following example shows the need of a new kind of dependency pairs.

Example 2. Consider the following TRS $\mathcal{R}$:

```
a -> c(f(a))
f(c(X)) -> X
```

together with $\mu(\mathtt{c}) = \varnothing$ and $\mu(\mathtt{f}) = \{1\}$. There is no μ-replacing subterm s in the right-hand sides of the rules which is rooted by a defined symbol. Thus, there is no 'regular' dependency pair. We could wrongly conclude that $\mathcal{R}$ is μ-terminating, which is not true:

[1] A symbol f is said to be *defined* in a TRS $\mathcal{R}$ if $\mathcal{R}$ contains a rule $f(l_1, \ldots, l_k) \to r$.

$$\texttt{f(a)} \hookrightarrow_\mu \underline{\texttt{f(c(f(a)))}}\ \texttt{f(a)} \hookrightarrow_\mu \cdots$$

Indeed, we must add the following dependency pair

```
F(c(X)) -> X
```

which would not be allowed in Arts and Giesl's approach [AG00] because the right-hand side is a variable.

After some preliminaries in Section 2, Section 3 introduces the general framework to compute and use context-sensitive dependency pairs for proving termination of *CSR*. The introduction of a new kind of dependency pairs (as in Example 2) leads to a new notion of context-sensitive dependency *chain*. We prove the correctness and completeness of the new approach, i.e., our dependency pairs approach fully characterize termination of *CSR*. We also show how to use term orderings for proving termination of *CSR* by means of the new approach. Furthermore, we are properly extending Arts and Giesl's approach: whenever $\mu(f) = \{1, \ldots, k\}$ for all k-ary symbols $f \in \mathcal{F}$, *CSR* and ordinary rewriting coincide; coherently, our results boil down into the standard results for the dependency pair approach. Section 4 shows how to compute the (estimated) context-sensitive dependency graph and investigates how to use term orderings together with the dependency graph to achieve automatic proofs of termination of *CSR* within the dependency pairs approach. Section 5 adapts Hirokawa and Middeldorp's subterm criterion [HM04] to *CSR*. Section 6 concludes.

2 Preliminaries

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote their concatenation as $p.q$. If p is a position, and Q is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. We denote the topmost position by Λ. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. We write $t \trianglerighteq s$ if $s = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright s$ if $t \trianglerighteq s$ and $t \neq s$. The symbol labelling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with zero or more 'holes' $\Box$ (a fresh constant symbol).

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined functions*, where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Context-sensitive rewriting. A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1, \ldots, ar(f)\}$ [Luc98]. Let $M_{\mathcal{F}}$ be the set of all

$\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F}, R)$). A binary relation R on terms is μ-monotonic if $t\,R\,s$ implies $f(t_1,\ldots,t_{i-1},t,\ldots,t_k)\,R\,f(t_1,\ldots,t_{i-1},s,\ldots,t_k)$ for all $f \in \mathcal{F}$, $i \in \mu(f)$, and $t, s, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F},\mathcal{X})$. The set of *$\mu$-replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i\in\mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i)$, if $t \notin \mathcal{X}$. The set of *replacing* variables of t is $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$. The μ-replacing subterm relation $\trianglerighteq_{\mu}$ is given by $t \trianglerighteq_{\mu} s$ if there is $p \in \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. We write $t \triangleright_{\mu} s$ if $t \trianglerighteq_{\mu} s$ and $t \neq s$. In *context-sensitive rewriting* (*CSR* [Luc98]), we (only) contract *replacing* redexes: t μ-rewrites to s, written $t \hookrightarrow_{\mu} s$ (or $t \hookrightarrow_{\mathcal{R},\mu} s$ and even $t \hookrightarrow s$), if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^{\mu}(t)$. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mu}$ is terminating. A term t is μ-terminating if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mu} t_2 \hookrightarrow_{\mu} \cdots \hookrightarrow_{\mu} t_n \hookrightarrow_{\mu} \cdots$ starting from t. A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a CS-TRS.

Dependency pairs. Given a TRS $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ a new TRS $\mathsf{DP}(\mathcal{R}) = (\mathcal{F}^{\sharp}, D(R))$ of *dependency pairs* for $\mathcal{R}$ is given as follows: if $f(t_1,\ldots,t_m) \to r \in R$ and $r = C[g(s_1,\ldots,s_n)]$ for some defined symbol $g \in \mathcal{D}$ and $s_1,\ldots,s_n \in \mathcal{T}(\mathcal{F},\mathcal{X})$, then $f^{\sharp}(t_1,\ldots,t_m) \to g^{\sharp}(s_1,\ldots,s_n) \in D(R)$, where $f^{\sharp}$ and $g^{\sharp}$ are new fresh symbols (called *tuple* symbols) associated to defined symbols f and g respectively [AG00]. Let $\mathcal{D}^{\sharp}$ be the set of tuple symbols associated to symbols in $\mathcal{D}$ and $\mathcal{F}^{\sharp} = \mathcal{F} \cup \mathcal{D}^{\sharp}$. As usual, for $t = f(t_1,\ldots,t_k) \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we write $t^{\sharp}$ to denote the *marked* term $f^{\sharp}(t_1,\ldots,t_k)$. Conversely, given a marked term $t = f^{\sharp}(t_1,\ldots,t_k)$, where $t_1,\ldots,t_k \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we write $t^{\natural}$ to denote the term $f(t_1,\ldots,t_k) \in \mathcal{T}(\mathcal{F},\mathcal{X})$. Given $T \subseteq \mathcal{T}(\mathcal{F},\mathcal{X})$, let $T^{\sharp}$ be the set $\{t^{\sharp} \mid t \in T\}$.

A reduction pair $(\succeq, \sqsupset)$ consists of a stable and weakly monotonic quasi-ordering $\succeq$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\succeq \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \succeq \subseteq \sqsupset$. Note that *monotonicity is not required* for $\sqsupset$.

3 Context-Sensitive Dependency Pairs

Let $\mathcal{M}_{\infty,\mu}$ be a set of minimal non-μ-terminating terms in the following sense: t belongs to $\mathcal{M}_{\infty,\mu}$ if t is non-μ-terminating and every strict *μ-replacing* subterm s of t (i.e., $t \triangleright_{\mu} s$) is μ-terminating. Obviously, if $t \in \mathcal{M}_{\infty,\mu}$, then $root(t)$ is a defined symbol. The following proposition establishes that, given a minimal non-μ-terminating term $t \in \mathcal{M}_{\infty,\mu}$, there are two ways for an infinite μ-rewrite sequence to proceed. The first one is by using 'visible' parts of the rules which correspond to μ-replacing subterms in the right-hand sides which are rooted by a defined symbol. The second one is by showing up 'hidden' non-μ-terminating subterms which are activated by *migrating* variables in a rule $l \to r$, i.e., variables $x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$ which are *not* μ-replacing in the left-hand side l but become μ-replacing in the right-hand side r.

Proposition 1. *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Then for all $t \in \mathcal{M}_{\infty,\mu}$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{M}_{\infty,\mu}$ such*

that $t \overset{>\Lambda}{\hookrightarrow}^* \sigma(l) \overset{\Lambda}{\to} \sigma(r) \trianglerighteq_\mu u$ *and either (1) there is a* μ*-replacing subterm* s *of* r *such that* $u = \sigma(s)$*, or (2) there is* $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$ *such that* $\sigma(x) \trianglerighteq_\mu u$*.*

Proposition 1 motivates the following.

Definition 1. *Let* $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ *be a TRS and* $\mu \in M_\mathcal{R}$*. We define* $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu) \cup \mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$ *to be the set of* context-sensitive *dependency pairs (CS-DPs) where:*

$$\mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu) = \{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s\}$$

and $\mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu) = \{l^\sharp \to x \mid l \to r \in R, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\}$*. We extend* $\mu \in M_\mathcal{F}$ *into* $\mu^\sharp \in M_{\mathcal{F}^\sharp}$ *by* $\mu^\sharp(f) = \mu(f)$ *if* $f \in \mathcal{F}$*, and* $\mu^\sharp(f^\sharp) = \mu(f)$ *if* $f \in \mathcal{D}$*.*

A rule $l \to r$ of a TRS $\mathcal{R}$ is μ-conservative if $\mathcal{V}ar^\mu(r) \subseteq \mathcal{V}ar^\mu(l)$, i.e., it does not contain migrating variables; $\mathcal{R}$ is μ-conservative if all its rules are (see [Luc06]). The following result is immediate from Definition 1.

Proposition 2. *If* $\mathcal{R}$ *is a* μ*-conservative TRS, then* $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$*.*

Therefore, in order to deal with μ-conservative TRSs $\mathcal{R}$ we only need to consider the 'classical' dependency pairs in $\mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$.

Example 3. Consider the TRS $\mathcal{R}$:

```
g(X) -> h(X)      h(d) -> g(c)      c -> d
```

together with $\mu(\mathtt{g}) = \mu(\mathtt{h}) = \varnothing$ [Zan97, Example 1]. $\mathsf{DP}(\mathcal{R}, \mu)$ is:

```
G(X) -> H(X)      H(d) -> G(c)
```

with $\mu^\sharp(\mathtt{G}) = \mu^\sharp(\mathtt{H}) = \varnothing$.

If the TRS $\mathcal{R}$ contains non-μ-conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

Example 4. Consider the TRS $\mathcal{R}$ [Zan97, Example 5]:

```
if(true,X,Y) -> X       f(X) -> if(X,c,f(true))
if(false,X,Y) -> Y
```

with $\mu(\mathtt{if}) = \{1, 2\}$. Then, $\mathsf{DP}(\mathcal{R}, \mu)$ is:

```
F(X) -> IF(X,c,f(true))        IF(false,X,Y) -> Y
```

with $\mu^\sharp(\mathtt{F}) = \{1\}$ and $\mu(\mathtt{IF}) = \{1, 2\}$.

Now we introduce the notion of chain of CS-DPs.

Definition 2 (Chain of CS-DPs). *Let* $(\mathcal{R}, \mu)$ *be a CS-TRS. Given* $\mathcal{P} \subseteq \mathsf{DP}(\mathcal{R}, \mu)$*, an* $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$*-chain is a finite or infinite sequence of pairs* $u_i \to v_i \in \mathcal{P}$*, for* $i \geq 1$ *such that there is a substitution* σ *satisfying both:*

1. $\sigma(v_i) \hookrightarrow^*_{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})$*, if* $u_i \to v_i \in \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$*, and*
2. *if* $u_i \to v_i = u_i \to x_i \in \mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$*, then there is* $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ *such that* $\sigma(x_i) \trianglerighteq_\mu s_i$ *and* $s_i^\sharp \hookrightarrow^*_{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})$*.*

for $i \geq 1$. Here, as usual we assume that different occurrences of dependency pairs do not share any variable (renamings are used if necessary).

An $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$-chain with $u_1 \to v_1 \in \mathcal{P}$ as heading dependency pair is called minimal *if $\sigma(u_1)^\natural \in \mathcal{M}_{\infty,\mu}$ and all dependency pairs in $\mathcal{P}$ occur infinitely often.*

Remark 1. When an $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain is written for a given substitution σ, we write $\sigma(u) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} \sigma(v)$ for steps which use a dependency pair $u \to v \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ but we rather write $\sigma(u) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} s^\sharp$ for steps which use a dependency pair $u \to x \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$, where s is as in Definition 2.

In the following, we use $\mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu)$ to denote the subset of dependency pairs in $\mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ whose migrating variables occur on non-μ-replacing immediate subterms in the left-hand side:

$$\mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu) = \{f^\sharp(u_1,\ldots,u_k) \to x \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) \mid \exists i, 1 \leq i \leq k, i \notin \mu(f^\sharp), x \in \mathcal{V}ar(u_i)\}$$

For instance, $\mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu) = \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ for the CS-TRS $(\mathcal{R},\mu)$ in Example 4. For this subset of CS-dependency pairs, we have the following.

Proposition 3. *There is no infinite $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$-chain with $\mathcal{P} \subseteq \mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu)$.*

The following result establishes the correctness of the context-sensitive depenency pairs approach.

Theorem 1 (Correctness). *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. If there is no infinite $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain, then $\mathcal{R}$ is μ-terminating.*

As an immediate consequence of Theorem 1 and Proposition 3, we have the following.

Corollary 1. *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. If $\mathsf{DP}(\mathcal{R},\mu) = \mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu)$, then $\mathcal{R}$ is μ-terminating.*

Example 5. Consider the following TRS $\mathcal{R}$ [Luc98, Example 15]

```
and(true,X) -> X          first(0,X) -> nil
and(false,Y) -> false     first(s(X),cons(Y,Z)) -> cons(Y,first(X,Z))
if(true,X,Y) -> X         from(X) -> cons(X,from(s(X)))
if(false,X,Y) -> Y
add(0,X) -> X
add(s(X),Y) -> s(add(X,Y))
```

with $\mu(\mathtt{cons}) = \mu(\mathtt{s}) = \mu(\mathtt{from}) = \varnothing$, $\mu(\mathtt{add}) = \mu(\mathtt{and}) = \mu(\mathtt{if}) = \{1\}$, and $\mu(first) = \{1,2\}$. Then, $\mathsf{DP}(\mathcal{R},\mu) = \mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu)$ is:

```
ADD(0,X) -> X          IF(true,X,Y) -> X
AND(true,X) -> X       IF(false,X,Y) -> Y
```

Thus, by Corollary 1 we conclude the μ-termination of $\mathcal{R}$.

Now we prove that the previous CS-dependency pairs approach is not only correct but also complete for proving termination of *CSR*.

Theorem 2 (Completeness). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. If $\mathcal{R}$ is μ-terminating, then there is no infinite $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain.*

Corollary 2 (Characterization of μ-termination). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. $\mathcal{R}$ is μ-terminating iff there is no infinite $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain.*

In the dependency pairs approach, the absence of infinite chains is checked by finding a *reduction pair* $(\succeq, \sqsupset)$ which is compatible with the rules and the dependency pairs [AG00]. In our setting, we can relax the monotonicity requirements and use *μ-reduction pairs* $(\gtrsim, \sqsupset)$ where $\gtrsim$ is a stable and μ-monotonic quasi-ordering which is compatible with the well-founded and stable ordering $\sqsupset$, i.e., $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. The following result shows how to use μ-reduction pairs for proving μ-termination. This is the context-sensitive counterpart of [AG00, Theorem 7]; however, a number of remarkable differences arise due to the treatment of the dependency pairs in $\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$. Basically, we need to ensure that the quasi-ordering is able to 'look' for a μ-replacing subterm inside the instantiation $\sigma(x)$ of a migrating variable x (hence we require $\trianglerighteq_\mu \subseteq \gtrsim$) and also connect a term which is rooted by defined symbol f and the corresponding dependency pair which is rooted by $f^\sharp$ (hence the requirement $f(x_1,\ldots,x_k) \gtrsim f^\sharp(x_1,\ldots,x_k)$).

Theorem 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$. Then, $\mathcal{R}$ is μ-terminating if and only if there is a μ-reduction pair $(\gtrsim, \sqsupset)$ such that,*

1. *$l \gtrsim r$ for all $l \to r \in R$,*
2. *$u \sqsupset v$ for all $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$, and*
3. *whenever $\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) \neq \varnothing$ we have that $\trianglerighteq_\mu \subseteq \gtrsim$, where $\trianglerighteq_\mu$ is the μ-replacing subterm relation on $\mathcal{T}(\mathcal{F},\mathcal{X})$, and*
   (a) *$u\ (\gtrsim \cup \sqsupset)\ v$ for all $u \to v \in \mathsf{DP}^1_{\mathcal{X}}(\mathcal{R},\mu)$, $u \sqsupset v$ for all $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) - \mathsf{DP}^1_{\mathcal{X}}(\mathcal{R},\mu)$, and $f(x_1,\ldots,x_k) \gtrsim f^\sharp(x_1,\ldots,x_k)$ for all $f \in \mathcal{D}$, or*
   (b) *$u\,(\gtrsim \cup \sqsupset)\,v$ for all $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ and $f(x_1,\ldots,x_k) \sqsupset f^\sharp(x_1,\ldots,x_k)$ for all $f \in \mathcal{D}$.*

4 Context-Sensitive Dependency Graph

As noticed by Arts and Giesl, the analysis of infinite sequences of dependency pairs can be made by looking at (the cycles $\mathfrak{C}$ of) the *dependency graph* associated to the TRS $\mathcal{R}$. The nodes of the dependency graph are the dependency pairs in $\mathsf{DP}(\mathcal{R})$; there is an arc from a dependency pair $u \to v$ to a dependency pair $u' \to v'$ if there are substitutions σ and θ such that $\sigma(v) \to^*_{\mathcal{R}} \theta(u')$.

Similarly, in the *context-sensitive (CS-)dependency graph*:

1. There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if there are substitutions σ and θ such that $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \theta(u')$.
2. There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ to *each* dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$.

Note that the use of $\mu^\sharp$ (which restricts reductions on the arguments of the dependency pair symbols $f^\sharp$) is essential: given a set of dependency pairs associated to a CS-TRS $(\mathcal{R},\mu)$, we have less arcs between them due to the presence of such replacement restrictions.

Example 6. Consider the CS-TRS in Example 1. $\mathsf{DP}(\mathcal{R},\mu)$ is:

```
F(a,b,X) -> F(X,X,X)
```

with $\mu^\sharp(\mathtt{F}) = \{3\}$. Although the dependency graph contains a cycle (due to $\sigma(\mathtt{F(X,X,X)}) \to^* \sigma(\mathtt{F(a,b,Y)})$ for $\sigma(X) = \sigma(Y) = \mathtt{c}$), the CS-dependency graph contains *no* cycle because it is *not* possible to $\mu^\sharp$-reduce $\theta(\mathtt{F(X,X,X)})$ into $\theta(\mathtt{F(a,b,Y)})$ for any substitution θ (due to $\mu^\sharp(\mathtt{F}) = \{3\}$).

As noticed by Arts and Giesl, the presence of an infinite chain of dependency pairs correspond to a cycle in the dependency graph (but not vice-versa).

Again, as an immediate consequence of Theorem 1 and Proposition 3, we have the following.

Corollary 3. *Let $\mathcal{R}$ be a TRS, $\mu \in M_\mathcal{R}$ and $\mathfrak{C} \subseteq \mathsf{DP}^1_\mathcal{X}(\mathcal{R},\mu)$ be a cycle. Then, there is no minimal $(\mathcal{R},\mathfrak{C},\mu^\sharp)$-chain.*

According to this, and continuing Example 6, we conclude the μ-termination of $\mathcal{R}$ in Example 1.

4.1 Estimating the CS-Dependency Graph

In general, the (context-sensitive) dependency graph of a TRS is *not* computable and we need to use some approximation of it. Following [AG00], we describe how to approximate the CS-dependency graph of a CS-TRS $(\mathcal{R},\mu)$. Let $\textsc{Cap}^\mu$ be given as follows: let D be a set of defined symbols (in our context, $D = \mathcal{D} \cup \mathcal{D}^\sharp$):

$$\textsc{Cap}^\mu(x) = x \text{ if } x \text{ is a variable}$$
$$\textsc{Cap}^\mu(f(t_1,\ldots,t_k)) = \begin{cases} y & \text{if } f \in D \\ f([t_1]^f_1,\ldots,[t_k]^f_1) & \text{otherwise} \end{cases}$$

where y is intended to be a new, fresh variable which has not yet been used and given a term s, $[s]^f_i = \textsc{Cap}^\mu(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. Let $\textsc{Ren}^\mu$ given by: $\textsc{Ren}^\mu(x) = y$ if x is a variable and $\textsc{Ren}^\mu(f(t_1,\ldots,t_k)) = f([t_1]^f_1,\ldots,[t_k]^f_k)$ for evey k-ary symbol f, where given a term $s \in \mathcal{T}^\sharp(\mathcal{F},\mathcal{X})$, $[s]^f_i = \textsc{Ren}^\mu(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. Then, we have an arc from $u_i \to v_i$ to $u_j \to v_j$ if $\textsc{Ren}^\mu(\textsc{Cap}^\mu(v_i))$ and u_j unify; following [AG00], we say that v_i and u_j are *μ-connectable*. The following result whose proof is similar to that of [AG00, Theorem 21] (we only need to take into account the replacement restrictions indicated by the replacement map μ) formalizes the correctness of this approach.

Proposition 4. *Let $(\mathcal{R},\mu)$ be a CS-TRS. If there is an arc from $u \to v$ to $u' \to v'$ in the CS-dependency graph, then v and u' are μ-connectable.*

Example 7. (Continuing Ex. 6) Since $\text{REN}^{\mu^\sharp}(\text{CAP}^{\mu^\sharp}(\texttt{F(X,X,X)})) = \texttt{F(X,X,Z)}$ and `F(a,b,Y)` do not unify, we conclude (and this can be easily implemented) that the CS-dependency graph for the CS-TRS $(\mathcal{R},\mu)$ in Example 1 has no cycle.

4.2 Checking μ-Termination with the Dependency Graph

For the cycles in the dependency graph, the absence of infinite chains is checked by finding (possibly different) *reduction pairs* $(\succeq_{\mathfrak{C}}, \sqsupset_{\mathfrak{C}})$ for each cycle $\mathfrak{C}$ [GAO02, Theorem 3.5]. In our setting, we use μ-reduction pairs.

Theorem 4 (Use of the CS-dependency graph). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$. Then, $\mathcal{R}$ is μ-terminating if and only if for each cycle $\mathfrak{C}$ in the context-sensitive dependency graph there is a μ-reduction pair $(\gtrsim_{\mathfrak{C}}, \sqsupset_{\mathfrak{C}})$ such that, $R \subseteq \gtrsim_{\mathfrak{C}}$, $\mathfrak{C} \subseteq \gtrsim_{\mathfrak{C}} \cup \sqsupset_{\mathfrak{C}}$, and*

1. *If $\mathfrak{C} \cap \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) = \varnothing$, then $\mathfrak{C} \cap \sqsupset_{\mathfrak{C}} \neq \varnothing$*
2. *If $\mathfrak{C} \cap \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) \neq \varnothing$, then $\trianglerighteq_\mu \subseteq \gtrsim_{\mathfrak{C}}$ (where $\trianglerighteq_\mu$ is the μ-replacing subterm relation on $\mathcal{T}(\mathcal{F},\mathcal{X})$), and*
   (a) $\mathfrak{C} \cap \sqsupset_{\mathfrak{C}} \neq \varnothing$ and $f(x_1,\ldots,x_k) \gtrsim_{\mathfrak{C}} f^\sharp(x_1,\ldots,x_k)$ for all $f^\sharp$ in $\mathfrak{C}$, or
   (b) $f(x_1,\ldots,x_k) \sqsupset_{\mathfrak{C}} f^\sharp(x_1,\ldots,x_k)$ for all $f^\sharp$ in $\mathfrak{C}$.

Following Hirokawa and Middeldorp, the practical use of Theorem 4 concerns the so-called *strongly connected components*(SCCs) of the dependency graph, rather than the cycles themselves (which are exponentially many) [HM04, HM05]. A strongly connected component in the (CS-)dependency graph is a *maximal cycle*, i.e., it is not contained in any other cycle. According to Hirokawa and Middeldorp, when considering an SCC $\mathfrak{C}$, we *remove* from $\mathfrak{C}$ those pairs $u \to v$ satisfying $u \sqsupset v$. Then, we recompute the SCCs with the remaining pairs in the CS-dependency graph and start again (see [HM05, Section 4]). In our setting, it is not difficult to see that, if the condition $f(x_1,\ldots,x_k) \sqsupset_{\mathfrak{C}} f^\sharp(x_1,\ldots,x_k)$ for all $f \in \mathcal{D}$ holds for a given cycle $\mathfrak{C}$, then we can remove from $\mathfrak{C}$ *all* dependency pairs in $\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$, thus continuing from $\mathfrak{C} - \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$.

Example 8. Consider the CS-TRS $(\mathcal{R},\mu)$ in Example 4 and $\mathsf{DP}(\mathcal{R},\mu)$:

```
F(X) -> IF(X,c,f(true))
IF(false,X,Y) -> Y
```

with $\mu^\sharp(\texttt{F}) = \{1\}$ and $\mu^\sharp(\texttt{IF}) = \{1,2\}$. These two CS-dependency pairs form the only cycle in the CS-dependency graph. The μ-reduction pair $(\geq, >)$ induced by the polynomial interpretation

$$\begin{array}{ccc} [\texttt{c}] = [\texttt{true}] = 0 & [\texttt{f}](x) = x & [\texttt{F}](x) = x \\ [\texttt{false}] = 1 & [\texttt{if}](x,y,z) = x+y+z & [\texttt{IF}](x,y,z) = x+z \end{array}$$

can be used to prove the μ-termination of $\mathcal{R}$.

The use of *argument filterings*, which is standard in the current formulations of the dependency pairs method, also adapts without changes to the context-sensitive setting. This is a simple consequence of [AG00, Theorem 11] (using μ-monotonicity instead of monotonicity for the quasi-orderings is not a problem).

5 Subterm Criterion

In [HM04], Hirokawa and Middeldorp introduce a very interesting *subterm criterion* which permits to ignore certain cycles of the dependency graph.

Definition 3. [HM04] *Let $\mathcal{R}$ be a TRS and $\mathfrak{C} \subseteq \mathsf{DP}(\mathcal{R})$ such that every dependency pair symbol in $\mathfrak{C}$ has positive arity. A* simple projection *for $\mathfrak{C}$ is a mapping π that assigns to every k-ary dependency pair symbol $f^\sharp$ in $\mathfrak{C}$ an argument position $i \in \{1,\dots,k\}$. The mapping that assigns to every term $f^\sharp(t_1,\dots,t_k) \in \mathcal{T}^\sharp(\mathcal{F},\mathcal{X})$ with $f^\sharp$ a dependency pair symbol in $\mathcal{R}$ its argument position $\pi(f^\sharp)$ is also denoted by π.*

In the following result, for a simple projection π and $\mathfrak{C} \subseteq \mathsf{DP}(\mathcal{R},\mu)$, we let $\pi(\mathfrak{C}) = \{\pi(u) \to \pi(v) \mid u \to v \in \mathfrak{C}\}$. Note that $u, v \in \mathcal{T}^\sharp(\mathcal{F},\mathcal{X})$, but $\pi(u), \pi(v) \in \mathcal{T}(\mathcal{F},\mathcal{X})$.

Theorem 5. *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. Let $\mathfrak{C} \subseteq \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ be a cycle. If there exists a simple projection π for $\mathfrak{C}$ such that $\pi(\mathfrak{C}) \subseteq \trianglerighteq_\mu$, and $\pi(\mathfrak{C}) \cap \triangleright_\mu \neq \varnothing$, then there is no minimal $(\mathcal{R},\mathfrak{C},\mu^\sharp)$-chain.*

Note that the result is restricted to cycles which do *not* include dependency pairs in $\mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$. The following result provides a kind of generalization of the subterm criterion to simple projections which only consider *non-μ-replacing* arguments of tuple symbols.

Theorem 6. *Let $\mathcal{R} = (\mathcal{F},R)$ be a TRS, $\mu \in M_\mathcal{F}$ and $\mathfrak{C} \subseteq \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ be a cycle. Let $\gtrsim$ be a stable quasi-ordering on terms whose strict and stable part $>$ is well-founded and π be a simple projection for $\mathfrak{C}$ such that for all $f^\sharp$ in $\mathfrak{C}$, $\pi(f^\sharp) \notin \mu^\sharp(f^\sharp)$ and $\pi(\mathfrak{C}) \subseteq \gtrsim$.*

1. *If $\mathfrak{C} \cap \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) = \varnothing$ and $\mathfrak{C} \cap > \neq \varnothing$, then there is no minimal $(\mathcal{R},\mathfrak{C},\mu^\sharp)$-chain.*
2. *If $\mathfrak{C} \cap \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) \neq \varnothing$, $\trianglerighteq_\mu \subseteq \gtrsim$ (where $\trianglerighteq_\mu$ is the μ-replacing subterm relation on $\mathcal{T}(\mathcal{F},\mathcal{X})$), and*
   (a) *$\mathfrak{C} \cap > \neq \varnothing$ and $f(x_1,\dots,x_k) \gtrsim x_{\pi(f^\sharp)}$ for all $f \in \mathcal{D}$ such that $f^\sharp$ is in $\mathfrak{C}$, or*
   (b) *$f(x_1,\dots,x_k) > x_{\pi(f^\sharp)}$ for all $f \in \mathcal{D}$ such that $f^\sharp$ is in $\mathfrak{C}$,*
   then there is no minimal $(\mathcal{R},\mathfrak{C},\mu^\sharp)$-chain.

Example 9. Consider the CS-TRS $(\mathcal{R},\mu)$ in Example 3. $\mathsf{DP}(\mathcal{R},\mu)$ is:

```
G(X) -> H(X)
H(d) -> G(c)
```

where $\mu^\sharp(\mathtt{G}) = \mu^\sharp(\mathtt{H}) = \varnothing$. The dependency graph contains a single cycle including both of them. The only simple projection is $\pi(\mathtt{G}) = \pi(\mathtt{H}) = 1$. Since $\pi(\mathtt{G(X)}) = \pi(\mathtt{H(X)})$, we only need to guarantee that $\pi(\mathtt{H(d)}) = \mathtt{d} > \mathtt{c} = \pi(\mathtt{G(c)})$ holds for a stable and well-founded ordering $>$. This is easily fulfilled by, e.g., a polynomial ordering.

6 Conclusions

We have shown how to use dependency pairs in proofs of termination of *CSR*. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of *CSR*. Some interesting differences arise which can be summarized as follows: in sharp contrast to the standard dependency pairs approach, where all dependency pairs have tuple symbols $f^\sharp$ both in the left- and right-hand sides, we have dependency pairs having a single *variable* in the right-hand side. These variables reflect the effect of the *migrating* variables into the termination behavior of *CSR*. This leads to a new definition of chain of context-sensitive dependency pairs which also differs from the standard approach in that we have to especially deal with such migrating variables. As in Arts and Giesl's approach, the presence or absence of infinite chains of dependency pairs from $\mathsf{DP}(\mathcal{R},\mu)$ characterizes the μ-terminaton of $\mathcal{R}$ (Theorems 1 and 2). Furthermore, we are also able to use term orderings to ensure the absence of infinite chains of context-sensitive dependency pairs (Theorem 3). In fact, we are properly extending Arts and Giesl's approach: whenever $\mu(f) = \{1,\ldots,k\}$ for all k-ary symbols $f \in \mathcal{F}$, *CSR* and ordinary rewriting coincide and all these results and techniques boil down into well-known results and techniques for the dependency pairs approach.

Regarding the practical use of the CS-dependency pairs in proofs of termination of *CSR*, we have shown how to build and use the corresponding CS-dependency graph to either prove that the rules of the TRS and the cycles in the CS-dependency graph are compatible with some reduction pair (Theorem 4) or to prove that there are cycles which do not need to be considered at all (Theorems 5 and 6). We have implemented these ideas as part of the termination tool MU-TERM [AGIL07, Luc04a]. We refer the reader to [AGIL07] for details about the practical impact of the techniques developed in this paper. ¿From this preliminary results, we can well conclude that the CS-dependency pairs can play in *CSR* the (practical and theoretical) role than dependency pairs play in rewriting.

There are many other aspects of the dependency pairs approach which are also worth to be considered and eventually extended to *CSR* (e.g., narrowing refinements, modularity issues, innermost computations, usable rules, ...). These aspects provide an interesting subject for future work.

References

[AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.

[AGIL07] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science, to appear*, 2007.

[BLR02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In *Proc. of CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.

[DLMMU06] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2006.

[GAO02] J. Giesl, T. Arts, and E. Ohlebusch Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation* 34(1):21-58, 2002.

[GL02] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In *Proc. of RULE'02*, pages 29-41, ACM Press, New York, 2002.

[GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In *Proc. of RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

[GM04] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379-427, 2004.

[GTS04] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In *Proc. of LPAR'04*, LNCS 3452:301-331, Springer-Verlag, Berlin, 2004.

[HM04] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In *Proc. of RTA'04*, LNCS 3091:249-268, Springer-Verlag, Berlin, 2004.

[HM05] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199:172-199, 2005.

[Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[Luc02] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.

[Luc04a] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting In *Proc. of RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at `http://www.dsic.upv.es/~slucas/csr/termination/muterm`.

[Luc04b] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In *Proc. of FOSSACS'04*, LNCS 2987:318-332, Springer-Verlag, Berlin 2004.

[Luc05] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.

[Luc06] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, to appear 2006.

[Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.

[Ter03] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.

[Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In *Proc. of RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.

18.2 Improving the Context-Sensitive Dependency Graph

2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph**. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 188 (2007) 91–103
www.elsevier.com/locate/entcs

Improving the Context-sensitive Dependency Graph

Beatriz Alarcón, Raúl Gutiérrez, and Salvador Lucas [1,2]

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain

Abstract

The dependency pairs method is one of the most powerful technique for proving termination of rewriting and it is currently central in most automatic termination provers. Recently, it has been adapted to be used in proofs of termination of context-sensitive rewriting. The use of *collapsing* dependency pairs i.e., *having a single variable in the right-hand side* is a novel and essential feature to obtain a correct framework in this setting. Unfortunately, dependency pairs behave as a kind of *glue* in the context-sensitive dependency graph which makes the cycles bigger, thus making some proofs of termination harder. In this paper we show that this effect can be safely mitigated by removing some arcs from the graph, thus leading to faster and easier proofs. Narrowing dependency pairs is also introduced and used here to eventually simplify the treatment of the context-sensitive dependency graph. We show the practicality of the new techniques with some benchmarks.

Keywords: Dependency pairs, term rewriting, program analysis, termination.

1 Introduction

Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. With *context-sensitive rewriting* (*CSR* [10,11]) we can *achieve* a terminating behavior with non-terminating Term Rewriting Systems (TRSs [14,15]), by pruning (all) infinite rewrite sequences. In *CSR* we only rewrite μ-replacing subterms. Here, μ is a *replacement map*, i.e., a mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f of the signature $\mathcal{F}$ [10]. We use them to discriminate the argument positions on which the rewriting steps are allowed. Then, t_i is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$; every term t (as a whole) is μ-replacing by definition. For other subterms we proceed inductively

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-0007, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

[2] Email: {balarcon,rgutierrez,slucas}@dsic.upv.es

1571-0661/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.entcs.2007.05.040

in this way. Then, for a given TRS, we obtain a restriction of rewriting which we call *context-sensitive rewriting*. Proving termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and programming languages (see [5,6,8,11,13] for further motivation).

The *dependency pairs method* [1] is one of the most powerful techniques for proving termination of rewriting. Roughly speaking, given a TRS $\mathcal{R}$, the dependency pairs associated to $\mathcal{R}$ conform a new TRS $\mathsf{DP}(\mathcal{R})$ which (together with $\mathcal{R}$) determines the so-called *dependency chains* whose finiteness characterizes termination of $\mathcal{R}$. The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. In [3], the dependency pairs method has been adapted to be used in proofs of termination of *CSR*. The technique has been implemented in the tool MU-TERM [2,12]. Basically, the non-variable subterms in the right-hand sides of the rules which are considered to build the CS-dependency pairs must be *μ-replacing* terms. Nevertheless such 'standard' dependency pairs do not suffice to obtain a correct method for proving termination of *CSR*.

Example 1.1 [3, Example 2] Consider the following TRS $\mathcal{R}$:

```
a -> c(f(a))
f(c(X)) -> X
```

together with $\mu(\mathtt{c}) = \varnothing$ and $\mu(\mathtt{f}) = \{1\}$. There is no μ-replacing subterm s in the right-hand sides of the rules which is rooted by a defined symbol. Thus, there is no 'standard' dependency pair. We could wrongly conclude that $\mathcal{R}$ is μ-terminating, which is not true:

$\mathtt{f}(\underline{\mathtt{a}}) \hookrightarrow_\mu \underline{\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a})))} \hookrightarrow_\mu \mathtt{f}(\underline{\mathtt{a}}) \hookrightarrow_\mu \cdots$

Indeed, as shown in [3], we must add the following dependency pair

```
F(c(X)) -> X
```

which would not be allowed in Arts and Giesl's approach [1] because the right-hand side is a variable. In this paper, we call *collapsing* to such kind of dependency pairs.

As in Arts and Giesl's approach, the analysis of infinite sequences of context-sensitive dependency pairs can be made by looking at (the cycles $\mathfrak{C}$ of) the *context-sensitive dependency graph* associated to the CS-TRS $\mathcal{R}$. The nodes of the dependency graph are the dependency pairs in $\mathsf{DP}(\mathcal{R}, \mu)$. A disappointing aspect of *collapsing* context-sensitive dependency pairs (as `F(c(X)) -> X` above) is that they are connected to *every other* dependency pair in the context-sensitive dependency graph [3]. Intuitively, this is because the variable `X` in the right-hand side of the dependency pair could be instantiated to anything, thus being potentially able to 'connect' to every other dependency pair.

In this paper, we show that we can restrict the number of outcoming links of collapsing dependency pairs to dependency pairs headed by the so-called *hidden* symbols which are defined symbols that occur in non-replacing positions in the right-hand sides of some rule in the TRS. This leads to a new definition of the context-sensitive dependency graph which greatly improves the performance of the original method.

B. Alarcón et al. / Electronic Notes in Theoretical Computer Science 188 (2007) 91–103 93

Example 1.2 Consider the following non-terminating TRS $\mathcal{R}$ which can be used to compute the list of prime numbers [7] :

```
primes -> sieve(from(s(0)))      tail(cons(X,Y)) -> Y
from(X) -> cons(X,from(s(X)))    if(true,X,Y) -> X
head(cons(X,Y)) -> X             if(false,X,Y) -> Y
filter(s(s(X)),cons(Y,Z)) ->
  if(divides(s(s(X)),Y),filter(s(s(X)),Z),cons(Y,filter(X,sieve(Y))))
sieve(cons(X,Y)) -> cons(X,filter(X,sieve(Y)))
```

together with $\mu(\texttt{cons}) = \mu(\texttt{if}) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for any other symbols f. No (automatic or manual) proof of termination for this CS-TRS has been reported to date. By using the dependency graph as defined in [3] we were not able to find a proof with MU-TERM 4.3 [2].

In contrast, with the new definition in this paper, we have *no* cycles! Thus, a direct (and automatic) proof of μ-termination of $\mathcal{R}$ is easy now.

Narrowing dependency pairs was also introduced by Arts and Giesl to improve the efficiency of the dependency pairs technique in proofs of termination [1]. Roughly speaking, under some conditions, a dependency pair can be replaced by a set of pairs which could simplify or restructure the dependency graph and eventually simplify the proof of termination. We also investigate this technique for dealing with the context-sensitive dependency graph.

After some preliminary definitions in Section 2, Section 3 introduces the notion of hidden symbol and investigates its properties in proofs of termination of *CSR*. Section 4 shows how to use it to improve the context-sensitive dependency graph. Section 5 adapts narrowing of dependency pairs to context-sensitive dependency pairs. Section 6 provides an experimental evaluation of our techniques. Section 7 concludes.

2 Preliminaries

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote its concatenation as $p.q$. If p is a position, and Q is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. We denote the empty chain by Λ. The set of positions of a term t is $\mathcal{P}os(t)$. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. We write $t \trianglerighteq s$ if $s = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright s$ if $t \trianglerighteq s$ and $t \neq s$. The symbol labelling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with zero or more 'holes' $\Box$ (a fresh constant symbol).

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side ($lhs$) of the rule is l and r is the right-hand side (rhs). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined functions*, where

$\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Context-sensitive rewriting.

A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1, \ldots, ar(f)\}$ [10]. Let $M_{\mathcal{F}}$ be the set of all $\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F}, R)$). A binary relation R on terms is μ-monotonic if $t\ R\ s$ implies $f(t_1, \ldots, t_{i-1}, t, \ldots, t_k)\, R\, f(t_1, \ldots, t_{i-1}, s, \ldots, t_k)$ for every $t, s, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. The set of *$\mu$-replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i)$, if $t \notin \mathcal{X}$. The set of *replacing* variables of t is $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$. The μ-replacing subterm relation $\trianglerighteq_{\mu}$ is given by $t \trianglerighteq_{\mu} s$ if there is $p \in \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. We write $t \triangleright_{\mu} s$ if $t \trianglerighteq_{\mu} s$ and $t \neq s$. In *context-sensitive rewriting* (*CSR* [10]), we (only) contract *replacing* redexes: t μ-rewrites to s, written $t \hookrightarrow_{\mu} s$ (or $t \hookrightarrow_{\mathcal{R},\mu} s$), if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^{\mu}(t)$. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mu}$ is terminating. A term t is μ-terminating if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mu} t_2 \hookrightarrow_{\mu} \cdots \hookrightarrow_{\mu} t_n \hookrightarrow_{\mu} \cdots$ starting from t. A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a CS-TRS.

Dependency pairs.

Given a TRS $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ a new TRS $\mathsf{DP}(\mathcal{R}) = (\mathcal{F}^{\sharp}, D(R))$ of *dependency pairs* for $\mathcal{R}$ is given as follows: if $f(t_1, \ldots, t_m) \to r \in R$ and $r = C[g(s_1, \ldots, s_n)]$ for some defined symbol $g \in \mathcal{D}$ and $s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, then $f^{\sharp}(t_1, \ldots, t_m) \to g^{\sharp}(s_1, \ldots, s_n) \in D(R)$, where $f^{\sharp}$ and $g^{\sharp}$ are new fresh symbols (called *tuple* symbols) associated to defined symbols f and g respectively [1]. Let $\mathcal{D}^{\sharp}$ be the set of tuple symbols associated to symbols in $\mathcal{D}$ and $\mathcal{F}^{\sharp} = \mathcal{F} \cup \mathcal{D}^{\sharp}$. As usual, for $t = f(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we write $t^{\sharp}$ to denote the *marked* term $f^{\sharp}(t_1, \ldots, t_k)$. Conversely, given a marked term $t = f^{\sharp}(t_1, \ldots, t_k)$, where $t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we write $t^{\flat}$ to denote the term $f(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Given $T \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, let $T^{\sharp}$ be the set $\{t^{\sharp} \mid t \in T\}$.

3 Structure of infinite μ-rewrite sequences

Let $\mathcal{M}_{\infty,\mu}$ be a set of minimal non-μ-terminating terms in the following sense: t belongs to $\mathcal{M}_{\infty,\mu}$ if t is non-μ-terminating and every strict *μ-replacing* subterm s of t (i.e., $t \triangleright_{\mu} s$) is μ-terminating. Obviously, if $t \in \mathcal{M}_{\infty,\mu}$, then $root(t)$ is a defined symbol. Furthermore, since μ-terminating terms are preserved under μ-rewriting, it follows that $\mathcal{M}_{\infty,\mu}$ is also preserved under *inner* μ-rewritings.

Lemma 3.1 *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t \in \mathcal{M}_{\infty,\mu}$. If $t \overset{>\epsilon}{\hookrightarrow}{}^{*} s$, then $s \in \mathcal{M}_{\infty,\mu}$.*

The following proposition establishes that, given $t \in \mathcal{M}_{\infty,\mu}$, there are two ways for an infinite μ-rewrite sequence to proceed. The first one is by using 'visible' parts of the rules which correspond to μ-replacing subterms in the right-hand sides which

are rooted by a defined symbol. The second one is by showing up 'hidden' non-μ-terminating subterms which are activated by *migrating* variables in a rule $l \to r$, i.e., variables $x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$ which are *not* μ-replacing in the left-hand side l but become μ-replacing in the right-hand side r.

Proposition 3.2 [3] *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. For all $t \in \mathcal{M}_{\infty,\mu}$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{M}_{\infty,\mu}$ such that $t \overset{>\epsilon}{\hookrightarrow}{}^{*} \sigma(l) \overset{\epsilon}{\to} \sigma(r) \trianglerighteq_{\mu} u$ and either (1) there is a μ-replacing subterm s of r such that $u = \sigma(s)$, or (2) there is $x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$ such that $\sigma(x) \trianglerighteq_{\mu} u$.*

Now we investigate the structure of such sequences in more detail. In the following, we write $t \rhd_{\not\mu} s$ to denote that s is a non-replacing (hence strict!) subterm of t: $t \rhd_{\not\mu} s$ if there is $p \in \mathcal{P}os(t) - Pos^{\mu}(t)$ such that $s = t|_p$.

Definition 3.3 [Hidden symbol] Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. We say that $f \in \mathcal{F}$ is a *hidden symbol* if there is a rule $l \to r \in R$ and $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $r \rhd_{\not\mu} t$ and $root(t) = f$. Let $\mathcal{H}(\mathcal{R}, \mu)$ (or just $\mathcal{H}$, if $\mathcal{R}$ and μ are clear for the context) be the set of all hidden symbols in $(\mathcal{R}, \mu)$.

Lemma 3.4 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution. If there is a rule $l \to r \in R$ such that $\sigma(l) \not\trianglerighteq t$ and $\sigma(r) \rhd_{\not\mu} t$, then there is no $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$. Furthermore, there is a term $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $r \rhd_{\not\mu} t'$, $\sigma(t') = t$ and $root(t) = root(t') \in \mathcal{H}$.*

Proof. By contradiction. If there is $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$, then since variables in l are always below some function symbol we have $\sigma(l) \rhd t$, leading to a contradiction.

Since there is no $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$ but we have that $\sigma(r) \rhd_{\not\mu} t$, then there is a non-variable and non-replacing position $p \in \mathcal{P}os_{\mathcal{F}}(r) - \mathcal{P}os^{\mu}(r)$, such that $root(r|_p) = root(t) \in \mathcal{H}(\mathcal{R}, \mu)$ and $\sigma(r|_p) = t$. Then, we let $t' = r|_p$. □

The following lemma establishes that minimal non-μ-terminating and non-μ-replacing subterms occurring in a μ-rewrite sequence involving only minimal terms directly come from the first term in the sequence or are rooted by a hidden symbol.

Lemma 3.5 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $\mathcal{A}$ be a finite μ-rewrite sequence $t_1 \hookrightarrow t_2 \hookrightarrow \cdots \hookrightarrow t_n$ with $t_i \in \mathcal{M}_{\infty,\mu}$ for all i, $1 \leq i \leq n$ and $n \geq 1$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_1 \not\trianglerighteq t$ and $t_n \rhd_{\not\mu} t$, then $root(t) \in \mathcal{H}$.*

Proof. By induction on n:

(i) If $n = 1$, then it is vacuously true.

(ii) If $n > 1$, then we assume that $t_1 \not\trianglerighteq t$ and $t_n \rhd_{\not\mu} t$. Let $l \to r \in R$ be such that $t_{n-1} = C[\sigma(l)]$ and $t_n = C[\sigma(r)]$ for some context $C[\,]$. We consider two cases: either $t_{n-1} \rhd_{\not\mu} t$ holds or not.
   (a) If $t_{n-1} \rhd_{\not\mu} t$, then by the induction hypothesis we have that $root(t) \in \mathcal{H}$.
   (b) If $t_{n-1} \rhd_{\not\mu} t$ does not hold, then one of the following cases holds:
      (1) $t_{n-1} \rhd_{\mu} t$; then $t_{n-1} \in \mathcal{M}_{\infty,\mu}$ implies that $t \notin \mathcal{M}_{\infty,\mu}$, leading to a contradiction.

(2) $t_{n-1} \not\trianglerighteq t$ (in particular, $\sigma(l) \not\trianglerighteq t$); then, since $t_n \rhd_{\not\mu} t$ there must be $\sigma(r) \rhd_{\not\mu} t$. Thus, by Lemma 3.4 we conclude that $root(t) \in \mathcal{H}$.

□

Now we use the previous lemmas to investigate infinite sequences that mix μ-rewriting steps on minimal non-μ-terminating terms and the extraction of such subterms as μ-replacing subterms of (instances of) right-hand sides of rules.

Proposition 3.6 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $\mathcal{A}$ be an infinite sequence of the form $t_1 \xrightarrow{\epsilon} s_1 \trianglerighteq_\mu t'_2 \overset{>\epsilon}{\hookrightarrow}{}^* t_2 \xrightarrow{\epsilon} s_2 \trianglerighteq_\mu t'_3 \overset{>\epsilon}{\hookrightarrow}{}^* t_3 \cdots$ with t_i, $t'_i \in \mathcal{M}_{\infty,\mu}$ for all $i \geq 1$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_i \rhd_{\not\mu} t$ for some $i \geq 1$, then $root(t) \in \mathcal{H} \cap \mathcal{D}$ or $t_1 \rhd_{\not\mu} t$.*

Proof. By induction on i:

(i) If $i = 1$, it is trivial.

(ii) If $i > 1$ and $t_i \rhd_{\not\mu} t$, then we consider two cases: either $t_{i-1} \rhd_{\not\mu} t$ holds or not.
  (a) If $t_{i-1} \rhd_{\not\mu} t$, then by the induction hypothesis we get $t_1 \rhd_{\not\mu} t$ or $root(t) \in \mathcal{H} \cap \mathcal{D}$, as desired.
  (b) If $t_{i-1} \rhd_{\not\mu} t$ does not hold, then let $l \to r \in R$ be such that $t_{i-1} = \sigma(l)$ and $s_{i-1} = \sigma(r) \trianglerighteq_\mu t'_i$. We consider two cases:
    (1) if $t_{i-1} \rhd_\mu t$ then being $t_{i-1} \in \mathcal{M}_{\infty,\mu}$ it would imply that $t \notin \mathcal{M}_{\infty,\mu}$, thus leading to a contradiction.
    (2) If $t_{i-1} \not\trianglerighteq t$, then we consider two cases: either $t'_i \rhd t$ or $t'_i \not\trianglerighteq t$.
      (A) If $t'_i \rhd t$, since $t'_i, t \in \mathcal{M}_{\infty,\mu}$ the case $t'_i \rhd_\mu t$ is excluded and the only possibility is that $t'_i \rhd_{\not\mu} t$. Then, since $\sigma(l) = t_{i-1} \not\trianglerighteq t$ and $\sigma(r) \trianglerighteq_\mu t'_i \rhd_{\not\mu} t$, i.e. $\sigma(r) \rhd_{\not\mu} t$, by Lemma 3.4 we conclude that $root(t) \in \mathcal{H}$. Since $t \in \mathcal{M}_{\infty,\mu}$, we have $root(t) \in \mathcal{H} \cap \mathcal{D}$
      (B) If $t'_i \not\trianglerighteq t$, then, by applying Lemma 3.1 and Lemma 3.5 to the μ-rewrite sequence $t'_i \overset{>\epsilon}{\hookrightarrow}{}^* t_i$ we conclude $root(t) \in \mathcal{H} \cap \mathcal{D}$.

□

As an immediate consequence of Proposition 3.6, we have the following result which we will use later.

Corollary 3.7 *Let $(\mathcal{R}, \mu)$ be a CS-TRS, $\mathcal{A}$ be an infinite sequence of the form $t_1 \xrightarrow{\epsilon} s_1 \trianglerighteq_\mu t'_2 \overset{>\epsilon}{\hookrightarrow}{}^* t_2 \xrightarrow{\epsilon} s_2 \trianglerighteq_\mu t'_3 \overset{>\epsilon}{\hookrightarrow}{}^* t_3 \cdots$ with t_i, $t'_i \in \mathcal{M}_{\infty,\mu}$ for all $i \geq 1$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_i \rhd_{\not\mu} t$ for some $i \geq 1$ and $root(t) \in \mathcal{D} - \mathcal{H}$, then $t_1 \rhd_{\not\mu} t$.*

4 Revised context-sensitive dependency graph

Proposition 3.2 motivates the definition of context-sensitive dependency pair(s) and chain of context-sensitive dependency pairs.

Definition 4.1 [CS-dependency pairs [3]] Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. We define $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ to be the set of

context-sensitive dependency pairs (CS-DPs) where:

$$\mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu) = \{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s\}$$

and $\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) = \{l^\sharp \to x \mid l \to r \in R, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\}$. We extend $\mu \in M_{\mathcal{F}}$ into $\mu^\sharp \in M_{\mathcal{F}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$.

Example 4.2 Consider the CS-TRS $(\mathcal{R},\mu)$ in Example 1.2. There are six context-sensitive dependency pairs:

```
1: PRIMES -> SIEVE(from(s(s(0))))
2: PRIMES -> FROM(s(s(0)))
3: TAIL(cons(X,Y)) -> Y
4: IF(true,X,Y) -> X
5: IF(false,X,Y) -> Y
6: FILTER(s(s(X)),cons(Y,Z)) ->
    IF(divides(s(s(X)),Y),filter(s(s(X)),Z),cons(Y,filter(X,sieve(Y))))
```

Note the three collapsing dependency pairs: (3), (4), and (5).

Definition 4.3 [Chain of CS-DPs [3]] Let $(\mathcal{R},\mu)$ be a CS-TRS. Given $\mathcal{P} \subseteq \mathsf{DP}(\mathcal{R},\mu)$, an $(\mathcal{R},\mathcal{P},\mu^\sharp)$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, for $i \geq 1$ such that there is a substitution σ satisfying both:

(i) $\sigma(v_i) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$, if $u_i \to v_i \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$, and

(ii) if $u_i \to v_i = u_i \to x_i \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$, then there is $s_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $\sigma(x_i) \trianglerighteq_\mu s_i$ and $s_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$.

Here, as usual we assume that different occurrences of dependency pairs do not share any variable (renamings are used if necessary). An $(\mathcal{R},\mathcal{P},\mu^\sharp)$-chain is called *minimal* if for all $i \geq 1$ $\sigma(u_i)^\natural \in \mathcal{M}_{\infty,\mu}$, $s_i \in \mathcal{M}_{\infty,\mu}$ (whenever they occur in the chain) and all dependency pairs in $\mathcal{P}$ occur infinitely often.

Remark 4.4 When an $(\mathcal{R},\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp)$-chain is written for a given substitution σ, we write $\sigma(u) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} \sigma(v)$ for steps which use a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$ but we rather write $\sigma(u) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} s^\sharp$ for steps which use a dependency pair $u \to x \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$, where s is as in Definition 4.3.

Theorem 4.5 (Correctness and completeness [3]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. $\mathcal{R}$ is μ-terminating if and only if there is no infinite $(\mathcal{R},\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp)$-chain.*

An essential aspect of the mechanization of the dependency pairs approach is the analysis of infinite sequences of dependency pairs by looking at (the cycles $\mathfrak{C}$ of) the *dependency graph* associated to the TRS $\mathcal{R}$. In [3],he *context-sensitive dependency graph*, is defined as follows:

(i) There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if there is a substitutions σ such that $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u')$.

(ii) There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ to *each* dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$.

Connecting each collapsing dependency pair with *every* other dependency pair

makes the cycles bigger, thus making some proofs of termination harder. Thanks to the results in the previous section, we can prove the following.

Theorem 4.6 *There is no infinite minimal $(\mathcal{R},\mathcal{P},\mu^\sharp)$-chain involving an infinite number of dependency pairs $u_i \to v_i \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ such that $root(u_{i+1})^\natural \notin \mathcal{H}$.*

Proof. By contradiction. Let A be an infinite $(\mathcal{R},\mathcal{P},\mu^\sharp)$-minimal chain of CS-DPs characterized by the CS-DPs $u_i \to v_i$ for $i \geq 1$:

$$\sigma(u_1) \overset{\epsilon}{\hookrightarrow}_\mathcal{P} s_1^\sharp \hookrightarrow^*_\mathcal{R} \sigma(u_2) \overset{\epsilon}{\hookrightarrow}_\mathcal{P} s_2^\sharp \hookrightarrow^*_\mathcal{R} \cdots$$

where, $s_i^\sharp = \sigma(v_i)$ if $u_i \to v_i \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ and $\sigma(x_i) \trianglerighteq_\mu s_i$ if $u_i \to v_i = u_i \to x_i \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$. Let I be the infinite set of indices satisfying that for all $i \in I$, $u_i \to v_i \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ and $root(u_{i+1})^\natural \notin \mathcal{H}$. Given $i \in I$, let $\eta(i)$ be the 'next' positive integer in I: $\eta(i) = min(\{j \in I \mid j > i\})$. Obviously, for all $i \in I$, $\eta(i) \in I$. Now consider the following sequence $A^\natural$ which is obtained from A by 'unsharping' the tuple symbols and using the rules $l_i \to r_i$ which originate the dependency pairs $u_i \to v_i$ which are used in A:

$$\sigma(u_1)^\natural \overset{\epsilon}{\hookrightarrow}_\mathcal{R} \sigma(r_1) \trianglerighteq_\mu s_1 \hookrightarrow^*_\mathcal{R} \sigma(u_2)^\natural \overset{\epsilon}{\hookrightarrow}_\mathcal{R} \sigma(r_2) \trianglerighteq_\mu s_2 \hookrightarrow^*_\mathcal{R} \cdots$$

which corresponds to A above: by minimality of A (see Definition 4.3), we have that $\sigma(u_i)^\natural, s_i \in \mathcal{M}_{\infty,\mu}$ for all $i \geq 1$. By definition of I, for all $i \in I$, $v_i = x_i \in \mathcal{X}$. By definition of collapsing dependency pair, $x_i \in \mathcal{P}os(u_i) - \mathcal{P}os^\mu(u_i)$ and $\sigma(x_i) \trianglerighteq_\mu s_i$. Thus, $\sigma(u_i) \rhd_{\not\mu} s_i$ for all $i \in I$. By repeatedly applying Corollary 3.7, we have that $\sigma(u_i) \rhd_{\not\mu} \sigma(u_{\eta(i)})$, i.e., $\sigma(u_i) \rhd \sigma(u_{\eta(i)})$ for all $i \in I$. Thus, we obtain an infinite $\rhd$-sequence which contradicts well-foundedness of $\rhd$. □

As a consequence of this result, we can dismiss the arcs of the dependency graph which connect collapsing dependency pairs $u \to v$ and dependency pairs $u' \to v'$ such that $root(u')^\natural \notin \mathcal{H}$. This leads to a new definition of the context-sensitive dependency graph:

Definition 4.7 [Context-Sensitive Dependency Graph] Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. The context-sensitive dependency graph consists of the set $\mathsf{DP}(\mathcal{R},\mu)$ of context-sensitive dependency pairs together with arcs which connect them as follows:

(i) There is an arc from a dependency pair $u \to v \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if there is a substitutions σ such that $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u')$.

(ii) There is an arc from a dependency pair $u \to v \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if $root(u')^\natural \in \mathcal{H}(\mathcal{R},\mu)$.

Example 4.8 Consider again the TRS $\mathcal{R}$ in Example 1.2. The hidden defined symbols are `filter`, `from` and `sieve`. The dependency graph which corresponds to this example is shown in Figure 1 (right). Note that, in contrast to the situation

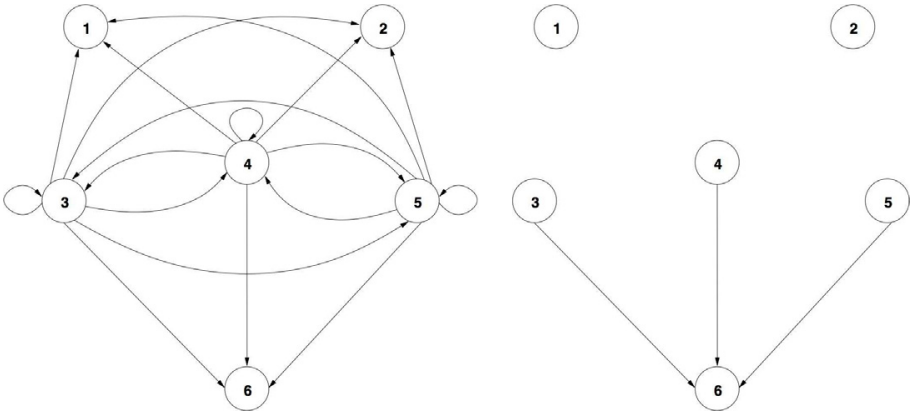


Fig. 1. Dependency graphs for Example 1.2: according to [3] (left) and according to Definition 4.7 (right)

with the old dependency graph (Figure 1, left) the new dependency graph has *no* cycle!

As noticed by Arts and Giesl, the presence of an infinite chain of dependency pairs corresponds to a cycle in the dependency graph (but not vice-versa). In the dependency graph this is true in the following sense: for each infinite chain of dependency pairs there is a suffix of the chain which corresponds to a cycle in the new dependency graph.

On the other hand, the treatment of cycles of the context-sensitive dependency graph for concluding termination by means of orderings remains as described in [3], but using the dependency graph in Definition 4.7.

Example 4.9 Consider the following TRS $\mathcal{R}$ [16, Example 4]

```
f(X) -> cons(X,f(g(X)))
g(0) -> s(0)
g(s(X)) -> s(s(g(X)))
sel(0,cons(X,Y)) -> X
sel(s(X),cons(Y,Z)) -> sel(X,Z)
```

with $\mu(\mathtt{0}) = \varnothing$, $\mu(\mathtt{f}) = \mu(\mathtt{g}) = \mu(\mathtt{s}) = \mu(\mathtt{cons}) = \{1\}$, and $\mu(\mathtt{sel}) = \{1, 2\}$. Then, $\mathsf{DP}(\mathcal{R}, \mu)$ is:

```
G(s(X)) -> G(X)
SEL(s(X),cons(Y,Z)) -> SEL(X,Z)
SEL(s(X),cons(Y,Z)) -> Z
```

The set of hidden symbols is $\mathcal{H} = \{\mathtt{f}, \mathtt{g}\}$ and there are two cycles:

(i) `G(s(X)) -> G(X)`

(ii) `SEL(s(X),cons(Y,Z)) -> SEL(X,Z)`

By using the subterm criterion [3, Section 5] we can easily prove that the system is μ-terminating.

5 Narrowing context-sensitive dependency pairs

There are examples where the automation of the CS-DP method can fail or be more difficult due to the *estimation* of the arcs that connect two CS-dependency pairs (by means of functions $\textsc{Cap}^\mu$ and $\textsc{Ren}^\mu$, see [3]).

Example 5.1 Consider the following example [13, Proposition 7]

```
f(0) -> cons(0,f(s(0)))
f(s(0)) -> f(p(s(0)))
p(s(X)) -> X
```

together with $\mu(\mathtt{f}) = \mu(\mathtt{p}) = \mu(\mathtt{s}) = \mu(\mathtt{cons}) = \{1\}$ and $\mu(\mathtt{0}) = \varnothing$. Then $\mathsf{DP}(\mathcal{R}, \mu)$ is:

```
F(s(0)) -> F(p(s(0)))
F(s(0)) -> P(s(0))
```

The *estimated* CS-dependency graph contains one cycle consisting of the CS-dependency pair

```
F(s(0)) -> F(p(s(0)))
```

However, this cycle does *not* belong to the CS-dependency graph because there is no way to μ-rewrite `F(p(s(0)))` into `F(s(0))`!

The problem is that with the estimated CS-dependency graph, we connect more dependency pairs than needed. The over-estimation eventually comes when a CS-dependency pair $u \to v$ is is connected to $u' \to v'$ in the estimated dependency graph and v and u' do not unify, i.e. at least a rewriting step with some rule of $\mathcal{R}$ is needed to reduce (some instance of) v to (the corresponding instance of) u'. It is then possible that, after performing such a necessary μ-rewriting step, the connection between them gets clearly lost, i.e, the nodes were not really connected in the graph. This is missed in the estimated dependency graph due to the use of $\textsc{Cap}^\mu$ and $\textsc{Ren}^\mu$. We can use *context-sensitive* narrowing to avoid this problem.

Definition 5.2 [Context-sensitive narrowing [10]] Let $(\mathcal{R}, \mu)$ be a CS-TRS. A term t μ-narrows to a term s (written $t \leadsto_\mu s$), if there exists a non-variable position $p \in \mathcal{P}os^\mu(t)$, θ is the most general unifier of $t|_p$ and l for a rewrite rule $l \to r$ in $\mathcal{R}$ (sharing no variable with t), and $s = \theta(t[r]_p)$.

To achieve more precision when connecting two CS-DPs in a $(\mathcal{R}, \mathsf{DP}(\mathcal{R}, \mu), \mu^\sharp)$-chain, we may perform all possible μ-narrowings steps on v in order to develop the reductions from (instances of) v to (instances of) u'.Then, we obtain new terms v_1, …, v_n which are μ-narrowings of v with unifier θ_i for $i \in \{1, \ldots, n\}$ and can be used instead of v. Not only the right-hand sides of the CS-dependency pairs are μ-narrowed: the unifier which used in the narrowing step should also be applied on the left-hand sides of the μ-narrowed pairs. Therefore, we can replace a CS-dependency pair $u \to v$ by all new μ-narrowed pairs $\theta_1(u) \to v_1, \ldots, \theta_n(u) \to v_n$. The next result shows that under those conditions, the set of CS-dependency pairs can be replaced by their narrowings without losing correctness or completeness.

Theorem 5.3 (Narrowing refinement for CS-termination) *Let $\mathcal{R}$ be a TRS and let $\mathcal{P}$ be a set of CS-dependency pairs. Let $u \to v \in \mathcal{P}$ such that v is linear and for all $u' \to v' \in \mathcal{P}$ (with renamed variables) the terms v and u' are not unifiable. Let*

$$\mathcal{P}' = (\mathcal{P} - \{u \to v\}) \cup \{u' \to v' \mid u' \to v' \text{ is a narrowing of } u \to v\}.$$

There exists an infinite $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$-chain iff there exists an infinite $(\mathcal{R}, \mathcal{P}', \mu^\sharp)$-chain.

Proof. The proof of this theorem corresponds to the proof of Theorem 25 in [1] . Note that only dependency pairs in $\mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ can be narrowed. As in Arts and Giesl's proof, requiring the no-unification between the CS-dependency pair to narrow and the rest of the set; the linearity of v; and the renaming of the variables of the different (occurrences of) dependency pairs is still necessary to guarantee that narrowing CS-dependency pairs do not miss any chain from $\mathcal{P}$. The main difference is that the reductions between dependency pairs are μ-reductions, but since we are using μ-narrowing, the whole proof is adapted without loss of generality. □

Thus, after narrowing the dependency pairs in $\mathsf{DP}(\mathcal{R}, \mu)$ we can build a *narrowed* dependency graph. Afterwards, we can use it to check termination as usual.

Example 5.4 (Continuing Example 5.1) Since the right-hand side of the CS-dependency pair in Example 5.1 does not unify with any left-hand side of a dependency pair, (including itself) and it can be μ-narrowed at position 1 (notice that $\mu(\mathtt{f})=\{1\}$) by using the rule

```
p(s(X)) -> X
```

we can replace it by its μ-narrowed CS-dependency pair:

```
F(s(0)) -> F(0)
```

The *narrowed* pair does not form any cycle in the estimated narrowed graph and termination is easily proved now.

6 Experiments

The techniques described in the previous sections have been implemented as part of the tool MU-TERM [2,12]. We have used our new implementation to compare with the last version of the tool: MU-TERM 4.3. The benchmarks were executed in a completely automatic way (see [2] for a description of MU-TERM's termination expert) and with a timeout of 1 minute on the 90 examples in the Context-Sensitive Rewriting subcategory of the 2006 Termination Competition, available through the URL:

http://www.lri.fr/~marche/termination-competition/2006

As remarked above, our termination expert works as explained in [2] for version 4.3 of MU-TERM. For the new version 4.4 of MU-TERM, we have just used the new definition of the (eventually narrowed) dependency graph. We have compared our new implementation with the previous version of MU-TERM (corresponding to [3]).

Termination Tool	Total	CS-DPs	CSRPO	Transf.	Average time
MU-TERM **(PROLE'06)**	66	65	0	1	1.68s
MU-TERM **(FST&TCS'06)**	56	45	7	4	4.55s
AProVE	56	0	0	56	4.74s

Table 1

We have also used AProVE for proving termination of the examples. AProVE [9] is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques. AProVE is also able to prove termination of Context-Sensitive Rewriting by using *transformations*. Such transformations obtain a proof of the μ-termination of a TRS $\mathcal{R}$ as a proof of termination of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation). If we are able to prove *termination* of $\mathcal{R}^{\mu}_{\Theta}$ (using the standard methods), then the μ-termination of $\mathcal{R}$ is ensured (see [13] for a recent survey).

A complete account of our experiments can be found here:

http://www.dsic.upv.es/~rgutierrez/muterm/prole/benchmarks.html

Table 1 summarizes our bechmarks. As shown in Table 1, the results make clear the advantages of the new refinement: we are able to prove 10 additional examples and the proofs are almost *three* times faster (in the average).

Furthermore, we can say that the new refinement developed for the CS-DP approach greatly improves on the use of other techniques: the use of transformations and other (also powerful) techniques like CSRPO [4] becomes now anecdotic or null.

7 Conclusions

We have introduced a simplification of the context-sensitive dependency graph by restricting the outcoming links of collapsing dependency pairs to dependency pairs headed by the so-called *hidden* symbols. Hidden symbols are defined symbols that occur in non-replacing positions in the right-hand sides of some rule in the TRS. This greatly improves the performance of termination proofs based on the dependency graph proposed in [3]. Narrowing context-sensitive dependency pairs has also been investigated. It can also be helpful to simplify or restructure the dependency graph and eventually simplify the proof of termination. Regarding the practical use of the (refinements on the) new CS-dependency graph in proofs of termination of *CSR*, we have implemented these ideas as part of the termination tool MU-TERM and we have obtained quite good results in terms of new examples which could be proved, and also regarding the time for achieving the proofs.

Since the state-of-the-art of DP-based techniques for proving termination of *CSR* which has been introduced in this paper corresponds to the developement of DPs in the late nineties, we can conclude that further improvements of CS-DPs will evolve in such a way that the CS-dependency pairs approach can play for *CSR* the (practical and theoretical) role than dependency pairs play in rewriting.

Many other aspects of the dependency pairs approach are also worth to be considered and extended to *CSR* (modularity issues, innermost computations, usable rules,...). They provide an interesting subject for future work.

References

[1] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.

[2] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science, to appear*, 2007.

[3] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In N. Garg and S. Arun-Kumar, editors *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.

[4] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.

[5] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Sestoft and N. Heintze, editors, *Proc. of PEPM'04*, pages 147-158, ACM Press, New York, 2004.

[6] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2006.

[7] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

[8] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379-427, 2004.

[9] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281-286, Springer-Verlag, Berlin, 2006. Available at http://www-i2.informatik.rwth-aachen.de/AProVE.

[10] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[11] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.

[12] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting In V. van Oostrom, editor, *Proc. of RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at http://www.dsic.upv.es/~slucas/csr/termination/muterm.

[13] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782-1846, 2006.

[14] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.

[15] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.

[16] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.

18.3 Proving Termination of Context-Sensitive Rewriting with MU-TERM

3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with** MU-TERM . *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 188 (2007) 105–115
www.elsevier.com/locate/entcs

Proving Termination of Context-Sensitive Rewriting with MU-TERM

Beatriz Alarcón, Raúl Gutiérrez,
José Iborra and Salvador Lucas[1,2]

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain

Abstract

Context-sensitive rewriting (CSR) is a restriction of rewriting which forbids reductions on selected arguments of functions. Proving termination of CSR is an interesting problem with several applications in the fields of term rewriting and programming languages. Several methods have been developed for proving termination of CSR. The new version of MU-TERM which we present here implements all currently known techniques. Furthermore, we show how to combine them to furnish MU-TERM with an *expert* which is able to automatically perform the termination proofs. Finally, we provide a first experimental evaluation of the tool.

Keywords: Context-sensitive rewriting, term rewriting, program analysis, termination.

1 Introduction

Restrictions of rewriting can eventually *achieve* termination of rewriting computations by pruning all infinite rewrite sequences issued from every term. However, such kind of improvements can be difficult to prove. Context-sensitive rewriting (*CSR* [17,18]) is a restriction of rewriting which is useful for describing semantic aspects of programming languages (e.g., Maude, OBJ2, OBJ3, or CafeOBJ) and analyzing termination of the corresponding programs (see [8,9,13,18,22] for further motivation). In *CSR*, a *replacement map* μ discriminates, for each symbol of the signature, the argument positions $\mu(f)$ on which rewritings are allowed. In this way, for a given Term Rewriting System (TRS), we obtain a restriction of the rewrite relation which

[1] This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-0007, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

[2] Email: {balarcon,rgutierrez,jiborra,slucas}@dsic.upv.es

1571-0661/$ – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.entcs.2007.05.041

we call *context-sensitive rewriting*. A TRS $\mathcal{R}$ together with a replacement map μ is often called a CS-TRS and written $(\mathcal{R}, \mu)$.

Proving termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and programming languages (see [22]). There are two main approaches to prove termination of a CS-TRS $(\mathcal{R}, \mu)$:

- *direct proofs* use adapted versions of term orderings such as RPOs and polynomial orderings to compare the left- and right-hand sides of the rules [4,11,20,21]; and
- *transformations* which obtain a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation). If we are able to prove *termination* of $\mathcal{R}^{\mu}_{\Theta}$ (using the standard methods), then termination of the CS-TRS is ensured (see [22] for a recent survey).

MU-TERM was the first tool implementing techniques for proving termination of *CSR* [19]. The tool is available here:

http://www.dsic.upv.es/~slucas/csr/termination/muterm

Nowadays, the tool AProVE [15] also accepts context-sensitive termination problems specified in the TPDB format [3]. However, AProVE's proofs of termination of *CSR* are based on using transformations (i.e., no direct proof method is currently available). The new version of MU-TERM which we present here implements all currently known techniques. The new contributions which we report in this paper are the following:

(i) We have implemented the *context-sensitive recursive path ordering* described in [4].

(ii) We have implemented the *context-sensitive dependency pairs approach* described in [2].

(iii) On the basis of recent theoretical and experimental results (see [22]), we have developed a termination expert for *CSR* which combines the different existing techniques for proving termination of *CSR* without any interaction with the user.

Finally, we want to mention that the Maude Termination Tool [9]:

http://www.lcc.uma.es/~duran/MTT

which transforms proofs of termination of Maude programs into proofs of termination of *CSR* uses MU-TERM's expert as an auxiliary tool.

We assume a basic knowledge about term rewriting, see [24] for missing definitions and more information. In Section 2 we briefly describe the new features which have been added to MU-TERM. Section 3 discusses the termination expert. Section 4 provides an experimental evaluation of the new version of MU-TERM. Section 5 concludes and discusses future work.

[3] See http://www.lri.fr/~marche/tpdb/format.html

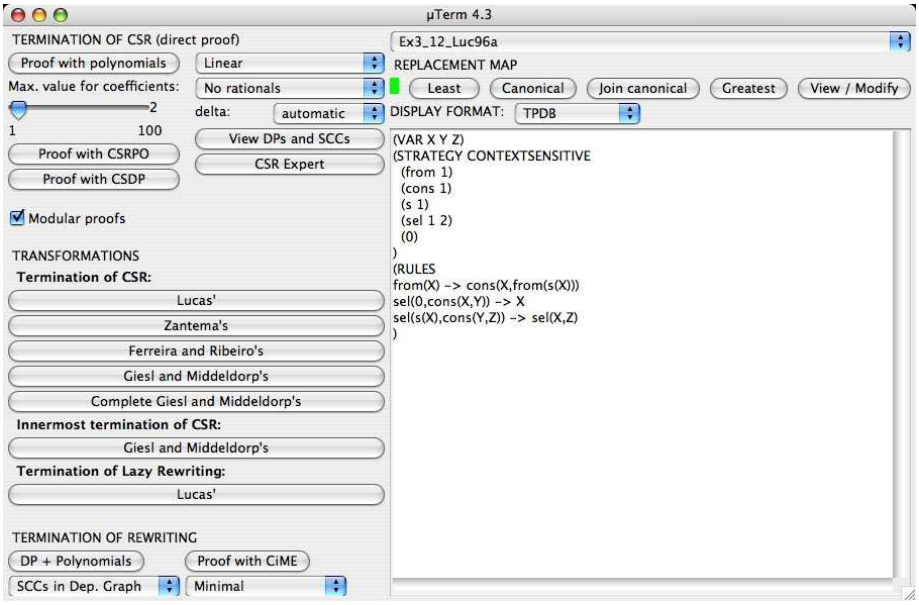


Fig. 1. Screenshot of the main window of MU-TERM 4.3

2 New features in MU-TERM

MU-TERM is written in Haskell [4], and wxHaskell [5] has been used to develop the graphical user interface. The system consists of more than 45 Haskell modules containing more than 14000 lines of code. Compiled versions in several platforms (Linux, Mac OSX, and Windows) and instructions for the installation are available on the MU-TERM WWW site. A recent hybrid (Haskell/C#) version of the tool is also available for the .NET platform [3].

MU-TERM has a graphical user interface (see Figure 1) whose details (menu structure, supported formats, etc.) are given in [19]. Let us briefly recall the main features of the tool.

- Modularity: If the *modular proofs* are activated, then MU-TERM attempts a *safe* decomposition of the TRS in such a way that the components satisfy the modularity requirements described in [10]. If it succeeds in performing a non-trivial decomposition (i.e., MU-TERM obtains more than one component), then individual proofs of termination of *CSR* are attempted for each component.
- Direct methods: MU-TERM implements the use of polynomial interpretations as described in [20,21]. An interesting feature of MU-TERM is that it generates polynomial interpretations with *rational* coefficients.
- Transformations: MU-TERM also implements a number of transformations for proving termination of *CSR* (see [13,22]).

[4] See http://haskell.org/.
[5] See http://wxhaskell.sourceforge.net.

In the following, we briefly describe the new features implemented in the current version of MU-TERM.

2.1 Context-Sensitive Recursive Path Ordering (CSRPO)

CSRPO extends the recursive path ordering (RPO [7]) to context-sensitive terms [4]. The first idea that comes in mind to extend RPO to *CSR* (CSRPO) is *marking* the symbols which are in blocked positions and consider them smaller than the active ones. Therefore, terms in blocked positions become smaller. However, marking all symbols in non-replacing positions can unnecessarily weaken the resulting ordering. Thus, in addition to the usual precedence [6] $\succeq_{\mathcal{F}}$ on the symbols of the signature $\mathcal{F}$ of the TRS, a *marking map*, denoted by $\mathtt{m}$, is also used. The marking map defines, for every symbol and every blocked position, the set of symbols that should be marked. By $\underline{\mathcal{F}}$ we denote the set of marked symbols corresponding to $\mathcal{F}$. Given a k-ary symbol f in $\mathcal{F} \cup \underline{\mathcal{F}}$ and $i \in \{1, \ldots, k\}$, a marking map $\mathtt{m}$ provides the subset of symbols in $\mathcal{F}$ that should be marked, i.e. $\mathtt{m}(f, i) \subseteq \mathcal{F}$. Marking maps are intended to mark *only* blocked arguments, i.e., $\mathtt{m}(f, i) = \varnothing$ if $i \in \mu(f)$ for all $f \in \mathcal{F}$. In this way, we mark only the necessary symbols (in blocked positions), see [4] for a thorough discussion.

Example 2.1 Consider the following TRS $\mathcal{R}$:

```
from(X) -> cons(X,from(s(X)))
sel(0,cons(X,Y)) -> X
sel(s(X),cons(Y,Z)) -> sel(X,Z)
```

together with $\mu(\mathtt{cons}) = \mu(\mathtt{s}) = \mu(\mathtt{from}) = \{1\}$ and $\mu(\mathtt{sel}) = \{1, 2\}$. The μ-termination of $\mathcal{R}$ can be proved by the CSRPO induced by the following precedence and marking map (computed by MU-TERM, see Figure 2):

$$\mathtt{sel} \succ_{\mathcal{F}} \mathtt{from} \succ_{\mathcal{F}} \mathtt{cons} \succ_{\mathcal{F}} \mathtt{s}$$

$$\mathtt{m}(\mathtt{cons}, 2) = \mathtt{m}(\underline{\mathtt{cons}}, 2) = \{\mathtt{from}\}, \quad \mathtt{m}(\underline{\mathtt{from}}, 1) = \varnothing$$

and lexicographic status for all function symbols.

Although the μ-termination of $\mathcal{R}$ in Example 2.1 can be proved by using the following polynomial interpretation:

$$[\mathtt{from}](X) = 3X + 2 \qquad [\mathtt{cons}](X, Y) = X + \tfrac{1}{3}Y \qquad [0] = 0$$
$$[\mathtt{s}](X) = 2X + 1 \qquad [\mathtt{sel}](X, Y) = 2X^2Y + X + Y + 1$$

the proof using CSRPO is much faster.

2.2 Context-Sensitive Dependency Pairs (CSDPs)

Recently, the *dependency pairs approach* [1], one of the most powerful techniques for proving termination of rewriting, has been generalized to be used in proofs of

[6] By a precedence, we mean a reflexive and transitive relation.

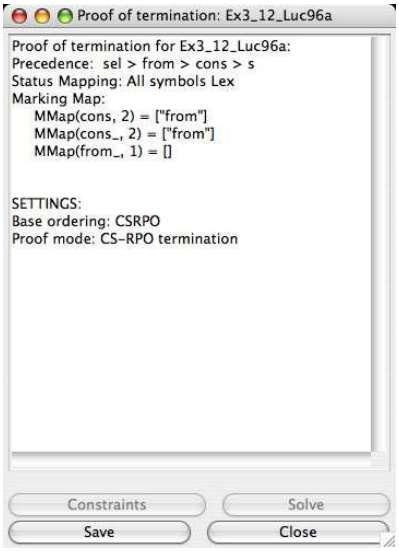


Fig. 2. Termination of *CSR* usingCSRPO

termination of *CSR* [2].

Roughly speaking, given a TRS $\mathcal{R}$, the dependency pairs $u \to v$ associated to $\mathcal{R}$ conform a new TRS $\mathsf{DP}(\mathcal{R})$ which (together with $\mathcal{R}$) determines the so-called *dependency chains* whose finiteness or infiniteness characterize termination of $\mathcal{R}$. The dependency pairs can be presented as a *dependency graph*, where the nodes of the graph are dependency pairs and the absence of infinite chains can be analyzed by considering the *cycles* in the graph. Two dependency pairs $u \to v$ and $u' \to v'$ in the graph are connected by an arc if there is a substitution σ which makes possible a (possibly empty) rewrite sequence (in $\mathcal{R}$) from $\sigma(v)$ to $\sigma(u')$. These ideas are generalized (with a number of non-trivial changes) to *CSR*.

Example 2.2 Consider the following non-terminating TRS $\mathcal{R}$ borrowing the well-known Toyama's example [12, Example 1]:

```
f(a,b,X) -> f(X,X,X)      c -> a      c -> b
```

together with $\mu(\mathtt{f}) = \{3\}$. The only dependency pair for this system is:

```
F(a,b,X) -> F(X,X,X)
```

where `F` is a 'marked' version (often called a *tuple symbol*) of `f` and we further assume that $\mu(\mathtt{F}) = \{3\}$. It is not difficult to see that there is no substitution σ which is able to originate a (possibly empty) *context-sensitive* rewrite sequence (with $\mathcal{R}$!) from $\sigma(\mathtt{F(X,X,X)})$ to $\sigma(\mathtt{F(a,b,X)})$. The replacement restriction $\mu(\mathtt{F}) = \{3\}$ is essential for this. Furthermore, this fact can be easily checked as explained in [2] and so it is implemented in MU-TERM.

A proof of μ-termination of $\mathcal{R}$ in Example 2.2 is *not* possible by using either CSRPO or polynomials with non-negative coefficients (see [11]). Also, as shown by Giesl and Middeldorp (see also [13]), among all the existing transformations for

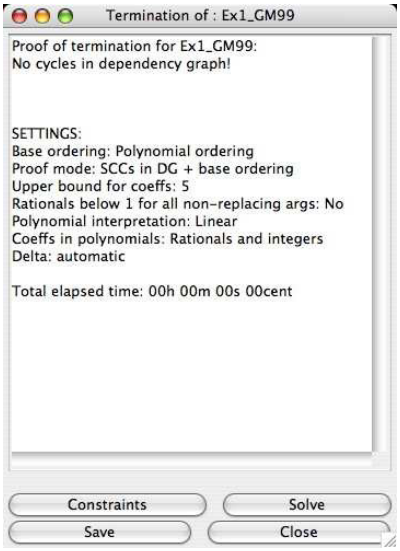


Fig. 3. Termination of *CSR* using dependency pairs

proving termination of *CSR*, only the *complete* Giesl and Middeldorp's transformation [13] (yielding a TRS $\mathcal{R}_C^\mu$) could be used in this case, but no concrete proof of termination for $\mathcal{R}_C^\mu$ is known yet. Furthermore, $\mathcal{R}_C^\mu$ has 13 dependency pairs and the dependency graph contains many cycles. In contrast, the CS-TRS has only *one* context-sensitive dependency pair and the corresponding dependency graph has *no* cycle! Thus, a direct and automatic proof of μ-termination of $\mathcal{R}$ is easy now (see Figure 3).

Although the subterms in the right-hand sides of the rules which are considered to build the context-sensitive dependency pairs are *μ-replacing* terms, considering *only non-variable subterms* (as in Arts and Giesl's approach [1]) is *not* sufficient to obtain a correct approximation. As discussed in [2], in general we also need to consider dependency pairs with *variables* in the right-hand sides.

Example 2.3 Consider the TRS $\mathcal{R}$ [26, Example 5]:

```
if(true,X,Y) -> X        f(X) -> if(X,c,f(true))
if(false,X,Y) -> Y
```

with $\mu(\texttt{if}) = \{1, 2\}$. There are two dependency pairs:

```
F(X) -> IF(X,c,f(true))
IF(false,X,Y) -> Y
```

with μ extended by $\mu(\texttt{F}) = \{1\}$ and $\mu(\texttt{IF}) = \{1, 2\}$.

A direct and automatic proof of μ-termination of $\mathcal{R}$ is possible with CSDPs by using an auxiliary polynomial ordering generated by a linear polynomial interpretation (computed by MU-TERM, see Figure 4).

A proof of μ-termination of $\mathcal{R}$ in Example 2.3 is *not* possible by using CSRPO. Furthermore, the μ-termination of $\mathcal{R}$ cannot be proved by using a polynomial or-

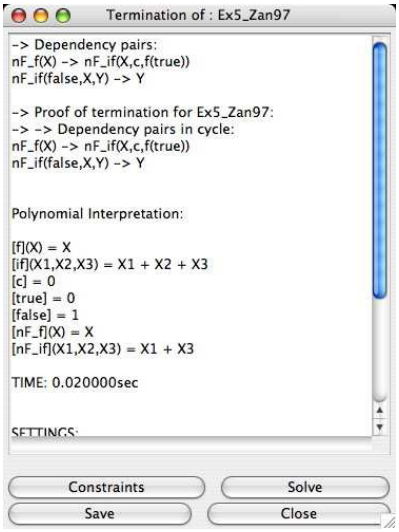


Fig. 4. Termination of *CSR* using dependency pairs and polynomials

dering based on a linear polynomial interpretation.

3 Automatically proving termination of *CSR* with MU-TERM

On the basis of recent theoretical and experimental results, we have developed a *termination expert* for *CSR* which combines the different existing techniques in a sequence of proof attempts which do not require any user interaction. The sequence of techniques which are tried by the expert is as follows:

(i) Context-sensitive dependency pairs with auxiliary polynomial orderings based on polynomial interpretations using either:
  (a) linear interpretations whose coefficients are taken from (1) $\{0,1\}$, (2) $\{0,1,2\}$, or (3) $\{0,\frac{1}{2},1,2\}$, in this order; or
  (b) simple-mixed interpretations linear whose coefficients are taken from (1) $\{0,1\}$, (2) $\{0,1,2\}$, or (3) $\{0,\frac{1}{2},1,2\}$, again in this order.

(ii) Context-sensitive recursive path ordering.

(iii) Polynomial orderings generated from either linear or simple-mixed polynomial interpretations whose coefficients are rational numbers of the form $\frac{p}{q}$ where $0 \leq p,q \leq 5$ and $q > 0$.

(iv) Transformations which obtain a TRS whose termination is proved by using the standard dependency pairs approach [1]. The transformations are attempted according to the decision tree in Figure 5 (explained below).

In the following, we motivate some of the choices we made for obtaining the concrete configuration of the previous sequence.

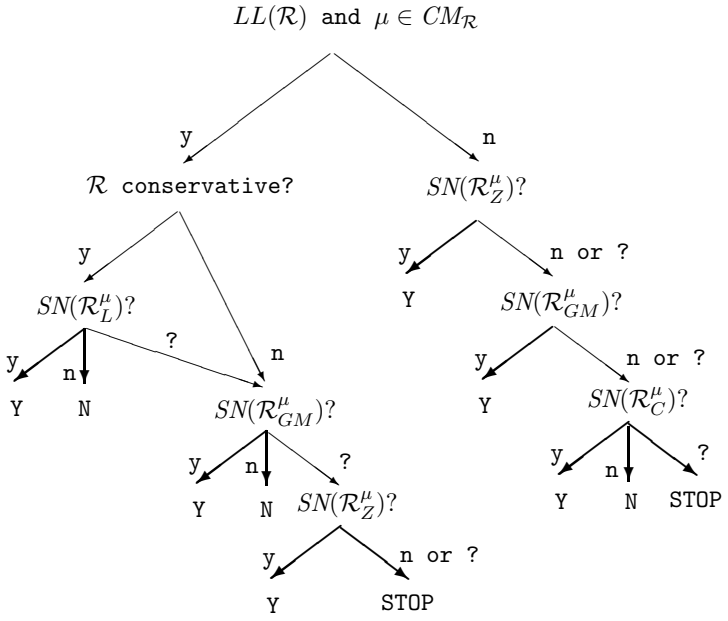


Fig. 5. Decision graph for proving termination of *CSR* by transformation

3.1 Use of polynomial interpretations

As shown in [21,23], the use of rational (or real) coefficients in polynomial interpretations can be helpful to achieve proofs of termination of (context-sensitive) rewriting. In this setting, in order to obtain a proof of μ-termination of a TRS $\mathcal{R} = (\mathcal{F}, R)$, we use *parametric* polynomial interpretations for the symbols $f \in \mathcal{F}$, whose indeterminate coefficients are intended to be *real* (or *rational*) instead of natural or integer numbers. The termination problem is rephrased as a set of polynomial constraints on the indeterminate coefficients. This set of constraints is intended to be solved *in the domain of the real numbers*. Although such polynomial constraints over the reals are decidable [25], the difficulty of the procedure depends on the degree and composition of the parametric polynomials that we use for this. As in [6], we consider classes of polynomials which are well-suited for automatization of termination proofs: linear and simple-mixed polynomial interpretations.

The automatic generation of rational coefficients can be computationally expensive. For instance, MU-TERM manages rational (nonnegative) coefficients $c \in \mathbb{Q}$ in polynomial interpretations as pairs numerator/denominator, i.e., $c = \frac{p}{q}$, where $p, q \in \mathbb{N}$ and $q > 0$. Thus, each rational coefficient c involves *two* integers. This leads to a huge search space in the corresponding constraint solving process [6,21]. For this reason, MU-TERM is furnished with three main *generation modes* [21]:

(i) *No rationals*: here, no rational coefficient is allowed.

(ii) *Rationals and integers*: here, since rational coefficients are intended to introduce non-monotonicity, we only use them with arguments $i \notin \mu(f)$.

(iii) *All rationals*: where all coefficients of polynomials are intended to be rational numbers.

These generation modes are orderly used by the expert to try different polynomial interpretations.

Regarding the *range* of the coefficients, we follow the usual practice in similar termination tools, where coefficients are bounded to take values 0, 1, or 2 (see [6,15,16,27]). Note that (as in those related tools) this choice is *heuristic*, usually based on the experience. We do not know of any theoretical or empirical investigation which tries to guide the choice of appropriate bounds for the coefficients depending on the concrete termination problem. From our side, we just added the value $\frac{1}{2}$ which enables a minimal (but still fruitful) use of rational coefficients. Again, these generation modes are orderly used by the expert to try different polynomial interpretations.

3.2 Use of transformations

In [22] we have investigated how to combine the different transformations for proving termination of *CSR*. Figure 5 provides a concrete decision tree for using the different transformations. Here, $LL(\mathcal{R})$ meas that $\mathcal{R}$ is left-linear, $CM_{\mathcal{R}}$ is the set of replacement maps which are not more restrictive than the *canonical* replacement map $\mu^{can}_{\mathcal{R}}$ of the TRS $\mathcal{R}$. This replacement map has a number of interesting properties (see [17,18]) and can be automatically computed for each TRS (for instance, the tool MU-TERM can do that) thus giving the user the possibility of using *CSR* without explicitly introducing hand-crafted replacement restrictions. Finally, $SN(\mathcal{R})?$ represeents a check of *termination* of the TRS $\mathcal{R}$. More details can be found in [22].

4 Experimental evaluation

As remarked in the introduction, besides MU-TERM, AProVE is currently the only tool which is able to prove termination of *CSR* by using (non-trivial) transformations. AProVE is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques. AProVE implements a termination expert which successively tries different transformations for proving termination of *CSR* and uses a variety of different and complementary techniques for proving termination of rewriting, see [15,14]. We have considered the (Linux-based, completely automatic) WST'06-version of AProVE and the set of 90 termination problems for *CSR* which have been used in the 2006 termination competition:

http://www.lri.fr/~marche/termination-competition/2006

A summary of the benchmarks can be found here:

http://www.dsic.upv.es/~rgutierrez/muterm/benchmarks.html

The benchmarks were executed on a PC equipped with an AMD Athlon XP processor at 2.4 GHz and 512 MB of RAM, running Linux (kernel 2.6.12). Both AProVE

and MU-TERM succeeded (running in a completely automatic way and with a time-out of 1 minute) on 56 examples; furthermore, the total elapsed time was almost the same for both tools. The MU-TERM expert used CSDPs in 45 of the 56 cases (80.4%); CSRPO in 7 cases (12.5%), and transformations in only 4 cases (7.1%, three of them using Zantema's transformation and one of them using Giesl and Middeldorp's incomplete transformation).

5 Conclusions and Future work

We have presented MU-TERM, a tool for proving termination of *CSR*. The tool has been improved with the implementation of new direct techniques for proving termination of *CSR* (the context-sensitive dependency pairs and the context-sensitive recursive path orderings) and an 'expert' for automatically proving termination of *CSR*. The new features perform quite well and have been shown useful in comparison with previously implemented techniques.

Future extensions of the tool will address the problem of efficiently using negative coefficients in polynomial interpretations (see [21] for further motivation). More research is also necessary to make the use of rational coefficients in proofs of termination much more efficient.

The current implementation of CSRPO is based on an ad-hoc incremental constraint solver which could be improved in many different directions. We plan to explore the reduction of the problem to a SAT-solving format, as described in [5]. We also plan to develop algorithms to solve polynomial constraints over the reals yielding exact (but not necessarily rational) solutions.

Finally, we want to improve the generation of reports and the inclusion of new, richer formats for input systems (e.g., Conditional TRSs, Many sorted TRSs, TRSs with AC symbols, etc.).

References

[1] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.

[2] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In N. Garg and S. Arun-Kumar, editors *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.

[3] B. Alarcón and S. Lucas. Building .NET GUIs for Haskell applications. In *Proc. of .NET'06*, pages 57-66, University of West-Bohemia, 2006.

[4] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.

[5] M. Codish, V. Lagoon, and P. Stuckey . Solving Partial Order Constraints for LPO Termination. In F. Pfenning, editor, *Proc. of 17th International Conference on Rewriting Techniques and Applications, RTA'06*, LNCS 4098, 2006.

[6] E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *Journal of Automated Reasoning*, 32(4):315-355, 2006.

[7] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[8] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Sestoft and N. Heintze, editors, *Proc. of ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147-158, ACM Press, New York, 2004.

[9] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2007.

[10] B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'02*, pages 50-61, ACM Press, New York, 2002.

[11] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In B. Fischer and E. Visser, editors, *Proc. of 3rd ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02*, pages 29-41, ACM Press, New York, 2002.

[12] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.

[13] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379-427, 2004.

[14] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors *Proc. of 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'04*, LNCS 3452:301-331, Springer-Verlag, Berlin, 2004.

[15] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281-286, Springer-Verlag, Berlin, 2006. Available at http://www-i2.informatik.rwth-aachen.de/AProVE.

[16] N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool. In J. Giesl, editor, *Proc. of 16th International Conference on Rewriting Techniques and Applications, RTA'05*, LNCS 3467:175-184, 2005.

[17] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.

[18] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.

[19] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting In V. van Oostrom, editor, *Proc. of 15h International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004.

[20] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In I. Walukiewicz, editor, *Proc. of 7th International Conference on Foundations of Software Science and Computation Structures, FOSSACS'04*, LNCS 2987:318-332, Springer-Verlag, 2004.

[21] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.

[22] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782-1846, 2006.

[23] S. Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 17(1):49-73, 2006.

[24] E. Ohlebusch. Advanced Topics in Term Rewriting. Springer-Verlag, Berlin, 2002.

[25] A. Tarski. A Decision Method for Elementary Algebra and Geometry. Second Edition. University of California Press, Berkeley, 1951.

[26] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.

[27] H. Zantema. TORPA: Termination of String Rewriting Systems. *Journal of Automated Reasoning*, 34:105-39, 2006.

18.4 Termination of Innermost Context-Sensitive Rewriting Using DPs

4. B. Alarcón, and S. Lucas. **Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs**. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, volume 4720 of *Lecture Notes in Artificial Intelligence*, pages 73–87, Springer-Verlag, 2007.

Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs*

Beatriz Alarcón and Salvador Lucas

DSIC, Universidad Politécnica de Valencia, Spain
{balarcon, slucas}@dsic.upv.es

Abstract. Innermost context-sensitive rewriting has been proved useful for modeling computations of programs of algebraic languages like Maude, OBJ, etc. Furthermore, innermost termination of rewriting is often easier to prove than termination. Thus, under appropriate conditions, a useful strategy for proving termination of rewriting is trying to prove termination of innermost rewriting. This phenomenon has also been investigated for context-sensitive rewriting (*CSR*). Up to now, only few transformations have been proposed and used to prove termination of innermost *CSR*. In this paper, we investigate direct methods for proving termination of innermost *CSR*. We adapt the recently introduced context-sensitive dependency pairs approach to innermost *CSR* and show that they can be advantageously used for proving termination of innermost *CSR*. We have implemented them as part of the termination tool MU-TERM.

1 Introduction

The *dependency pairs method* [3] is one of the most powerful techniques for proving termination of Term Rewriting Systems (TRSs [21,22]). Roughly speaking, given a TRS $\mathcal{R}$, the dependency pairs associated with $\mathcal{R}$ form a new TRS $\mathsf{DP}(\mathcal{R})$ which (together with $\mathcal{R}$) determines the so-called *dependency chains* which characterize termination of $\mathcal{R}$. The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. In [1], the dependency pairs method has been adapted for proving termination of *context-sensitive rewriting* (*CSR* [15,18]). With *CSR* we can *achieve* a terminating behavior for non-terminating TRSs by pruning (all) infinite rewrite sequences. In *CSR* we only rewrite μ-replacing subterms. Here, μ is a *replacement map*, i.e., a mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f of the signature $\mathcal{F}$ [15]. We use them to indicate the argument positions on which the rewriting steps are allowed. Then, t_i is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$; every term t (as a whole) is μ-replacing by definition.

* This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-2007, and the Generalitat Valenciana under grant GV06/285. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

B. Konev and F. Wolter(Eds.): FroCos 2007, LNCS 4720, pp. 73–87, 2007.
© Springer-Verlag Berlin Heidelberg 2007

For other subterms we proceed inductively in this way. Then, for a given TRS $\mathcal{R}$ and a replacement map μ, we obtain a restriction of rewriting which we call *context-sensitive rewriting*. A pair $(\mathcal{R}, \mu)$ is often called a context-sensitive TRS (CS-TRS). Proving termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and programming languages (see [20] for further motivation). Furthermore, termination of *innermost CSR* (i.e., the variant of *CSR* where only the deepest μ-replacing redexes are contracted) has proved useful for proving termination of programs in programming languages like Maude and OBJ* which permit to control the program execution by means of such context-sensitive annotations [16,17].

Proving innermost termination of rewriting is often easier than proving termination of rewriting [3] and, for some relevant classes of TRSs, innermost termination of rewriting is even equivalent to termination of rewriting [10,11]. In [7,12] it is proved that the equivalence between termination of innermost *CSR* and termination of *CSR* holds in some interesting cases (e.g., for *orthogonal* CS-TRSs).

Example 1. Consider the following orthogonal TRS $\mathcal{R}$ which is a variant of an example in [4]:

```
from(X) -> cons(X,from(s(X)))
sel(0,cons(X,XS)) -> X
sel(s(N),cons(X,XS)) -> sel(N,XS)
minus(X,0) -> X
minus(s(X),s(Y)) -> minus(X,Y)
quot(0,s(Y)) -> 0
quot(s(X),s(Y)) -> s(quot(minus(X,Y),s(Y)))
zWquot(nil,nil) -> nil
zWquot(cons(X,XS),nil) -> nil
zWquot(nil,cons(X:XS)) -> nil
zWquot(cons(X,XS),cons(Y,YS))->cons(quot(X,Y),zWquot(XS,YS))
```

together with $\mu(\texttt{cons}) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for all other symbols f. According to [6], innermost μ-termination of $\mathcal{R}$ implies its μ-termination as well. We will show how $\mathcal{R}$ can easily be proved innermost μ-terminating (and hence μ-terminating) by using the results in this paper.

In this paper, we extend the context-sensitive dependency pairs approach in [1] for proving termination of innermost *CSR*. Actually, techniques for proving termination of innermost *CSR* have already been investigated [7,16]. However, these papers only consider *transformational* techniques, where the original CS-TRS $(\mathcal{R}, \mu)$ is transformed into a TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation which has been used) whose *innermost* termination implies the innermost termination of *CSR* for $(\mathcal{R}, \mu)$. Up to now, no direct method has been proposed to prove termination of innermost *CSR*. As shown in [2], proofs of termination using context-sensitive dependency pairs (CSDPs) are much more powerful and faster than any other technique for proving termination of *CSR*. Dealing with innermost *CSR*, we have a similar situation.

Example 2. Consider the following TRS $\mathcal{R}$:

```
b -> c(b)
f(c(X),X) -> f(X,X)
```

together with $\mu(\mathtt{f}) = \{1, 2\}$ and $\mu(\mathtt{c}) = \varnothing$. This system is *not* μ-terminating:

$\mathtt{f(\underline{b},b)} \hookrightarrow \underline{\mathtt{f(c(b),b)}} \hookrightarrow \mathtt{f(\underline{b},b)} \hookrightarrow \ldots$

where $\hookrightarrow$ denotes a context-sensitive rewriting step. However $\mathcal{R}$ is innermost μ-terminating. We can give a very easy automatic proof of this fact because the *innermost* context-sensitive dependency graph has *no cycle*. In contrast, by using available transformations for proving innermost termination of *CSR* (see [8] for a survey), we could not obtain a proof by using tools (like AProVE or TTT) supporting innermost termination proofs (of rewriting).

In the dependency pairs approach, proofs of termination of innermost rewriting are easier than proofs of termination of rewriting because: (1) the estimated innermost dependency graph is more accurate, (2) it is possible to limit the attention on the so-called *usable* rules of the TRS (for a given cycle).

After some preliminaries in Section 2, in Section 3 we prove that termination of innermost *CSR* can be characterized by using an appropriate definition of chain of CSDPs. In Section 4 we show how to prove automatically innermost termination of *CSR* by using the *innermost context-sensitive dependency graph*. Section 5 adapts the notion of usable rules to deal with innermost *CSR*. Section 6 provides a first experimental evaluation of our techniques.

2 Preliminaries

Terms. Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote their concatenation as $p.q$. Positions are ordered by the standard prefix ordering $\leq$. If p is a position, and Q is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. We denote the topmost position by Λ. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of non-variable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$ while $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. We write $t \trianglerighteq s$ if $s = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright s$ if $t \trianglerighteq s$ and $t \neq s$. The symbol labelling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with zero or more 'holes' $\Box$ (a fresh constant symbol).

Term rewriting. A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined functions*, where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Context-sensitive rewriting. A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1,\dots,ar(f)\}$ [15]. Let $M_\mathcal{F}$ be the set of all $\mathcal{F}$-maps (or $M_\mathcal{R}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F},R)$). A binary relation R on terms is μ-monotonic if $t\,R\,s$ implies $f(t_1,\dots,t_{i-1},t,\dots,t_k)\,R\,f(t_1,\dots,t_{i-1},s,\dots,t_k)$ for all $f \in \mathcal{F}$, $i \in \mu(f)$, and $t,s,t_1,\dots,t_k \in \mathcal{T}(\mathcal{F},\mathcal{X})$. The set of *$\mu$-replacing positions* $\mathcal{P}os^\mu(t)$ of $t \in \mathcal{T}(\mathcal{F},\mathcal{X})$ is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i\in\mu(root(t))} i.\mathcal{P}os^\mu(t|_i)$, if $t \notin \mathcal{X}$. The set of *μ-replacing* variables of t is $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^\mu(t), t|_p = x\}$. The μ-replacing subterm relation $\trianglerighteq_\mu$ is given by $t \trianglerighteq_\mu s$ if there is $p \in \mathcal{P}os^\mu(t)$ such that $s = t|_p$. We write $t \triangleright_\mu s$ if $t \trianglerighteq_\mu s$ and $t \neq s$. In *context-sensitive rewriting* (*CSR* [15]), we (only) contract *μ-replacing* redexes: s μ-rewrites to t, written $s \hookrightarrow_\mu t$ (or $s \hookrightarrow_{\mathcal{R},\mu} t$ and even $s \hookrightarrow t$, if $\mathcal{R}$ and μ are clear from the context), if $s \xrightarrow{p}_\mathcal{R} t$ and $p \in \mathcal{P}os^\mu(s)$. A μ-normal form is a term which cannot be μ-rewritten. Let $\mathsf{NF}_\mu(\mathcal{R})$ (or just NF_μ if no confusion arises) be the set of μ-normal forms of a TRS $\mathcal{R}$. A μ-innermost redex is a redex t whose μ-replacing subterms are μ-normal forms: $t = \sigma(l)$ for some substitution σ and rule $l \to r \in \mathcal{R}$ and for all $p \in \mathcal{P}os^\mu(t)$, $t|_p \in \mathsf{NF}_\mu$. A term s innermost μ-rewrites to t, written $s \overset{i}{\hookrightarrow} t$, if $s \xrightarrow{p}_\mathcal{R} t$, $p \in \mathcal{P}os^\mu(s)$, and $s|_p$ is an μ-innermost redex. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_\mu$ is terminating. A term t is μ-terminating if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_\mu t_2 \hookrightarrow_\mu \cdots \hookrightarrow_\mu t_n \hookrightarrow_\mu \cdots$ starting from t. A TRS $\mathcal{R}$ is innermost μ-terminating if $\overset{i}{\hookrightarrow}_\mu$ is terminating. We write $s \overset{i}{\hookrightarrow}^!_\mathcal{R} t$ if $s \overset{i}{\hookrightarrow}^*_\mathcal{R} t$ and $t \in \mathsf{NF}_\mu$.

A pair $(\mathcal{R},\mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_\mathcal{R}$ is often called a CS-TRS.

Reduction pairs. A reduction pair $(\succeq,\sqsupset)$ consists of a stable and weakly monotonic quasi-ordering $\succeq$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\succeq \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \succeq \subseteq \sqsupset$. Note that *monotonicity is not required* for $\sqsupset$.

3 Termination of Innermost *CSR* with Dependency Pairs

In the following definition, given a term $t = f(t_1,\dots,t_k) \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we write $t^\sharp$ to denote the *marked* term $f^\sharp(t_1,\dots,t_k)$, where $f^\sharp$ is a new fresh symbol (called *tuple* symbol [3]). Given a signature $\mathcal{F}$, we let $\mathcal{F}^\sharp$ be the extension of $\mathcal{F}$ containing all tuple symbols for $\mathcal{F}$: $\mathcal{F}^\sharp = \mathcal{F} \cup \{f^\sharp \mid f \in \mathcal{F}\}$. Similarly, if $t = f^\sharp(t_1,\dots,t_k)$ is a marked term, we write $t^\natural$ to denote the unmarked term $f(t_1,\dots,t_k)$.

Definition 1 (CS-dependency pairs [1]). *Let $\mathcal{R} = (\mathcal{F},R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{R}$. We define* $\mathsf{DP}(\mathcal{R},\mu) = \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu) \cup \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ *to be the set of* context-sensitive *dependency pairs (CS-DPs) where:*

$$\mathsf{DP}_\mathcal{F}(\mathcal{R},\mu) = \{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s\}$$

and $\mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) = \{l^\sharp \to x \mid l \to r \in R, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\}$. *We extend $\mu \in M_\mathcal{F}$ into $\mu^\sharp \in M_{\mathcal{F}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$.*

Example 3. Consider the CS-TRS $(\mathcal{R}, \mu)$ in Example 2. There is only one context-sensitive dependency pair:

```
F(c(X),X) -> F(X,X)
```

with $\mu^\sharp(\mathtt{F}) = \{1, 2\}$.

In the CS-DP approach, termination of *CSR* is characterized as the absence of infinite chains of CS-DPs [1, Definition 2]. In innermost *CSR*, we only perform reduction steps on *innermost replacing redexes*. Therefore, we have to restrict the definition of chains in order to obtain an appropriate notion corresponding to innermost *CSR*. Regarding innermost reductions, arguments of a redex should be in *normal form* before the redex is contracted and, regarding *CSR*, the redex to be contracted has to be in a *replacing* position.

Definition 2 (Innermost μ-chain). *Given a CS-TRS $(\mathcal{P}, \mu^\sharp)$ of CS-DPs associated to a CS-TRS $(\mathcal{R}, \mu)$, an innermost $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$-chain is a sequence of pairs $u_j \to v_j \in \mathcal{P}$ such that there is a substitution σ such that $\sigma(u_j) \in \mathsf{NF}_\mu(\mathcal{R})$ and such that, for all $j \geq 1$,*

1. *$\sigma(v_j) \overset{i}{\hookrightarrow}^!_{\mathcal{R},\mu^\sharp} \sigma(u_{j+1})$, if $u_j \to v_j \in \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$, and*
2. *if $u_j \to v_j = u_j \to x_j \in \mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$, then there is some $s_j \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\sigma(x_j) \trianglerighteq_\mu s_j$ and $s_j^\sharp \overset{i}{\hookrightarrow}^!_{\mathcal{R},\mu^\sharp} \sigma(u_{j+1})$.*

As usual we assume that different occurrences of dependency pairs do not share any variables (renamings are used if necessary). An innermost $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$-chain is minimal *if for all $u_j \to v_j \in \mathcal{P}$ and $j \geq 1$, $\sigma(v_j)$ is innermost μ-terminating (whenever $u_j \to v_j \in \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$) and $s_j^\sharp$ is innermost μ-terminating (whenever $u_j \to v_j \in \mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$).*

Theorem 1. *A CS-TRS $(\mathcal{R}, \mu)$ is innermost μ-terminating if and only if no infinite minimal innermost μ-chain exists.*

Let $\mathcal{M}_{\infty,\mu}$ be a set of minimal non-μ-terminating terms in the following sense [1]: t belongs to $\mathcal{M}_{\infty,\mu}$ if t is non-μ-terminating and every strict *μ-replacing* subterm s of t (i.e., $t \rhd_\mu s$) is μ-terminating. The proof of this result uses the following result.

Proposition 1. [1, Proposition 1] *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{R}$. Then for all $t \in \mathcal{M}_{\infty,\mu}$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{M}_{\infty,\mu}$ such that $t \overset{>\Lambda}{\hookrightarrow}^* \sigma(l) \overset{\Lambda}{\to} \sigma(r) \trianglerighteq_\mu u$ and either (1) there is a μ-replacing subterm s of r such that $u = \sigma(s)$, or (2) there is $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$ such that $\sigma(x) \trianglerighteq_\mu u$.*

Proof. (of Theorem 1) We prove the *if* part by contradiction. We show that for any infinite innermost μ-rewriting sequence we can construct an infinite innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R}, \mu), \mu^\sharp)$-chain. Let innermost μ-rewriting below the root be $\overset{\geq i}{\hookrightarrow} = (\overset{>\Lambda}{\hookrightarrow} \cap \overset{i}{\hookrightarrow})$. If $\mathcal{R}$ is not innermost μ-terminating, then, by Proposition 1,

there is a term $t \in \mathcal{M}_{\infty,\mu}$, a rule $l \to r \in R$, a substitution σ, and a term $u \in \mathcal{M}_{\infty,\mu}$ such that $t \overset{>i}{\hookrightarrow} \sigma(l) \overset{\Lambda}{\to} \sigma(r) \trianglerighteq_\mu u$ (where every immediate replacing subterm of $\sigma(l)$ is a μ-normal form), u is not innermost μ-terminating and either

1. there is a μ-replacing subterm s of r such that $u = \sigma(s)$, or
2. there is $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$ such that $\sigma(x) \trianglerighteq_\mu u$.

In the first case above, we have a dependency pair $l^\sharp \to s^\sharp \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$ such that $u = \sigma(s) \in \mathcal{M}_{\infty,\mu}$, i.e., we can start an innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain beginning with $\sigma(l^\sharp) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} \sigma(s^\sharp)$. Note that $\sigma(l^\sharp) \in \mathsf{NF}_\mu(\mathcal{R})$.

In the second case above, since $u \in \mathcal{M}_{\infty,\mu}$, there is a rule $\lambda \to \rho$ such that $u \overset{>i}{\hookrightarrow}{}^! \sigma(\lambda)$ (since we can assume that the variables in this rule do not occur in l, we can use the same –conveniently extended– substitution σ) and $\sigma(\rho)$ contains a subterm in $\mathcal{M}_{\infty,\mu}$. Hence, $u^\sharp \overset{i}{\hookrightarrow}{}^!_{\mathcal{R},\mu^\sharp} \sigma(\lambda^\sharp)$. Furthermore, there is a dependency pair $l^\sharp \to x \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$ such that $\sigma(x) \trianglerighteq_\mu u$; thus, according to Definition 2 we can start an $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain beginning with

$$\sigma(l^\sharp) \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} u^\sharp$$

and then continuing with a dependency pair $u' \to v'$ such that $u' = \lambda^\sharp$ and $u^\sharp \overset{i}{\hookrightarrow}{}^!_{\mathcal{R},\mu^\sharp} \sigma(u')$. Note that $\sigma(l^\sharp), \sigma(\lambda^\sharp) \in \mathsf{NF}_\mu(\mathcal{R})$.

Thus, in both cases we can start an innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain which could be infinitely extended in a similar way by starting from $u^\sharp$. This contradicts our initial assumption.

On the other hand, in order to show that the criterion is also *necessary* for innermost termination of context-sensitive rewriting we assume that there exists an infinite innermost μ-chain that implies the existence of an infinite innermost μ-rewrite sequence. If there is an infinite innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain, then there is a substitution σ and dependency pairs $u_i \to v_i \in \mathsf{DP}(\mathcal{R},\mu)$ such that considering the first dependency pair $u_1 \to v_1$ in the sequence:

1. If $u_1 \to v_1 \in \mathsf{DP}_\mathcal{F}(\mathcal{R},\mu)$, then $v_1^\natural$ is a μ-replacing subterm of the right-hand-side r_1 of a rule $l_1 \to r_1$ in $\mathcal{R}$. Therefore, $r_1 = C_1[v_1^\natural]_{p_1}$ for some $p_1 \in \mathcal{P}os^\mu(r_1)$ and, since $\sigma(u_1) \in \mathsf{NF}_\mu$, we can perform the innermost μ-rewriting step $t_1 = \sigma(u_1^\natural) \overset{i}{\hookrightarrow}_{\mathcal{R},\mu} \sigma(r_1) = \sigma(C_1)[\sigma(v_1^\natural)]_{p_1} = s_1$, where $\sigma(v_1^\natural)^\sharp = \sigma(v_1) \overset{i}{\hookrightarrow}{}^!_{\mathcal{R},\mu^\sharp} \sigma(u_2)$ and $\sigma(u_2)$ also initiates an infinite innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^\sharp)$-chain. Note that $p_1 \in \mathcal{P}os^\mu(s_1)$.
2. If $u_1 \to x \in \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu)$, then there is a rule $l_1 \to r_1$ in $\mathcal{R}$ such that $u_1 = l_1^\sharp$, and $x \in \mathcal{V}ar^\mu(r_1) - \mathcal{V}ar^\mu(l_1)$, i.e., $r_1 = C_1[x]_{q_1}$ for some $q_1 \in \mathcal{P}os^\mu(r_1)$. Furthermore, since there is a subterm s such that $\sigma(x) \trianglerighteq_\mu s$ and $s^\sharp \overset{i}{\hookrightarrow}{}^!_{\mathcal{R},\mu^\sharp} \sigma(u_2)$, we can write $\sigma(x) = C_1'[s]_{p_1'}$ for some $p_1' \in \mathcal{P}os^\mu(\sigma(x))$. Therefore, since $\sigma(u_1) = \sigma(l_1)^\sharp \in \mathsf{NF}_\mu$, we can perform the innermost μ-rewriting step $t_1 = \sigma(l_1) \overset{i}{\hookrightarrow}_{\mathcal{R},\mu} \sigma(r_1) = \sigma(C_1)[C_1'[s]_{p_1'}]_{q_1} = s_1$ where $s^\sharp \overset{i}{\hookrightarrow}{}^!_{\mathcal{R},\mu^\sharp} \sigma(u_2)$ (hence

$s \overset{i}{\hookrightarrow}^{!} u_2^{\sharp}$) and $\sigma(u_2)$ initiates an infinite innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R},\mu), \mu^{\sharp})$-chain. Note that $p_1 = q_1.p_1' \in \mathcal{P}os^{\mu}(s_1)$.

Since $\mu^{\sharp}(f^{\sharp}) = \mu(f)$, and $p_1 \in \mathcal{P}os^{\mu}(s_1)$, we have that $s_1 \overset{i}{\hookrightarrow}^{!}_{\mathcal{R},\mu} t_2[\sigma(u_2)]_{p_1} = t_2$ and $p_1 \in \mathcal{P}os^{\mu}(t_2)$. Therefore, we can build in that way an infinite μ-rewrite sequence

$$t_1 \overset{i}{\hookrightarrow}_{\mathcal{R},\mu} s_1 \overset{i}{\hookrightarrow}^{!}_{\mathcal{R},\mu} t_2 \overset{i}{\hookrightarrow}_{\mathcal{R},\mu} \cdots$$

which contradicts the innermost μ-termination of $\mathcal{R}$.

Example 4. Consider again the CS-TRS $\mathcal{R}$ in Example 2. As shown in Example 3, there is only one CS-DP:

```
F(c(X),X) -> F(X,X)
```

Since $\mu^{\sharp}(\mathtt{F}) = \{1,2\}$, if a substitution σ satisfies $\sigma(\mathtt{F(c(X),X)}) \in \mathsf{NF}_{\mu}(\mathcal{R})$, then $\sigma(X) = s$ is in μ-normal form. Assume that the dependency pair is part of an innermost CS-DP-chain. Since there is no way to μ-rewrite $F(s,s)$, there must be $F(s,s) = F(c(t),t)$ for some term t, which means that $s = t$ and $c(t) = s$, i.e., $t = c(t)$ which is not possible. Thus, there is no infinite innermost chain of CS-DPs for $\mathcal{R}$, which is proved innermost terminating by Theorem 1.

Of course, ad-hoc reasonings like in Example 4 do not lead to automation. In the following section we discuss how to prove termination of innermost *CSR* by giving *constraints* on terms that can be solved by using standard methods.

4 Checking Innermost μ-Termination Automatically

The analysis of infinite sequences of dependency pairs can be handled by looking at (the cycles $\mathfrak{C}$ of) the dependency graph associated to the TRS $\mathcal{R}$ [3].

The Innermost Context-Sensitive dependency graph of a TRS $\mathcal{R}$ is the directed graph whose nodes are the CS-dependency pairs; there is an arc from $u \to v$ to $u' \to v'$ if $u \to v$, $u' \to v'$ is an innermost μ-chain.

In [2] we have investigated the structure of context-sensitive sequences in order to improve the CS-dependency graph. A μ-rewrite sequence can proceed in two ways: by means of *visible* parts of the rules that is, μ-replacing subterms in the right-hand sides which are rooted by a defined symbol, or showing up *hidden* non-μ-terminating subterms which are activated by *migrating* variables of a rule $l \to r$, i.e. those variables that are not μ-replacing in l and become μ-replacing in r.

Definition 3 (Hidden symbol [2]). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. We say that $f \in \mathcal{F}$ is a* hidden symbol *if there is a rule $l \to r \in R$ where f occurs at a non-μ-replacing position. Let $\mathcal{H}(\mathcal{R},\mu)$ (or just $\mathcal{H}$, if $\mathcal{R}$ and μ are clear from the context) be the set of all hidden symbols in $(\mathcal{R},\mu)$.*

Obviously, as innermost μ-chains are restricted chains (also restricted μ-chains!), the Innermost Context-Sensitive dependency graph (in the following ICSDG or ICS-dependency graph for short) is a subgraph of the dependency graph.

Definition 4 (Innermost CSDG). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. The innermost context-sensitive dependency graph consists of the set $\mathsf{DP}(\mathcal{R},\mu)$ of context-sensitive dependency pairs and arcs which connect them as follows:*

1. *There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if there are subtitutions σ and θ such that $\sigma(v) \overset{i}{\hookrightarrow}^{!} \theta(u')$ and $\sigma(u)$ and $\theta(u') \in \mathsf{NF}_{\mu}(\mathcal{R})$.*
2. *There is an arc from a dependency pair $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ to a dependency pair $u' \to v' \in \mathsf{DP}(\mathcal{R},\mu)$ if $root(u')^{\sharp} \in \mathcal{H}(\mathcal{R},\mu)$ and u and $u' \in \mathsf{NF}_{\mu}(\mathcal{R})$.*

Example 5. Consider the following CS-TRS $\mathcal{R}$ in [6]:

```
f(g(b)) -> f(g(a))        f(a) -> f(a)        a -> b
```

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{g}) = \varnothing$. Then $\mathsf{DP}(\mathcal{R},\mu)$ is:

```
F(g(b)) -> F(g(a))        F(a) -> F(a)        F(a) -> A
```

and $\mu^{\sharp}(\mathtt{F}) = \{1\}$. The CSDG contains a single cycle $\{$`F(a) -> F(a)`$\}$. However, the ICSDG is empty.

4.1 Approximating the ICSDG

In order to automatically build the Innermost Context-Sensitive Dependency Graph it is necessary to approximate it since for two dependency pairs $u \to v$ and $u' \to v'$ it is undecidable to know if there exist two substitutions σ and θ such that $\sigma(v)$ μ-reduces innermost to $\theta(u')$ and $\sigma(u)$ and $\theta(u')$ are instantiated to μ-normal forms. For this reason, we have to approximate the graph by computing a supergraph containing it in the same way as previous approaches [3,1]. In the context-sensitive setting, we have adapted functions Cap and Ren to be applied only on *μ-replacing* subterms [1]. On the other hand, in the innermost setting it is not necessary to use Ren since all variables are always instantiated to normal forms and cannot be reduced and $\textsc{Cap}(v)$ substitutes every subterm with a defined root symbol by fresh variables only if the term is not equal to subterms of u. To approximate the ICS-dependency graph, however, we have to combine both of them: we use $\textsc{Cap}^{\mu}_{u}(v)$ to replace all μ-replacing subterm rooted with a defined symbol whenever the term was not equal to a μ-replacing subterm of the left-hand side of the dependency pair u. We use $\textsc{Ren}^{\mu}_{u}(v)$ to replace by fresh variables those ones that are replacing in v but not in u since they are not μ-normalized. Given a term u, we let $\textsc{Cap}^{\mu}_{u}$ be given as follows: let D be a set of defined symbols (in our context, $D = \mathcal{D} \cup \mathcal{D}^{\sharp}$):

$$\textsc{Cap}^{\mu}_{u}(x) = x \text{ if } x \text{ is a variable}$$
$$\textsc{Cap}^{\mu}_{u}(f(t_1,\ldots,t_k)) = \begin{cases} y & \text{if } f \in D \\ f([t_1]^f_1,\ldots,[t_k]^f_k) & \text{otherwise} \end{cases}$$

where y is a new, fresh variable which has not yet been used and given a term s, $[s]^f_i = \textsc{Cap}^{\mu}_{u}(s)$ if $i \in \mu(f)$ and s is not equal to a μ-replacing subterm of u and $[s]^f_i = s$ otherwise. Given a term u, we let $\textsc{Ren}^{\mu}_{u}$ be given by: $\textsc{Ren}^{\mu}_{u}(x) = y$ if x is a variable and $\textsc{Ren}^{\mu}_{u}(f(t_1,\ldots,t_k)) = f([t_1]^f_1,\ldots,[t_k]^f_k)$ for evey k-ary symbol

f, where given a term $s \in \mathcal{T}^\sharp(\mathcal{F},\mathcal{X})$, $[s]_i^f = \text{REN}_u^\mu(s)$ if $i \in \mu(f)$ and the variable is not μ-replacing in u and $[s]_i^f = s$ otherwise.

We have an arc from $u \to v$ to $u' \to v'$ in the ICS-dependency graph if $\text{REN}_u^\mu(\text{CAP}_u^\mu(v))$ and u' are unifiable by some mgu σ such that $\sigma(u), \sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$; following [3], we say that v and u' are *innermost μ-connectable*. The following result whose proof is similar to that of [3, Theorem 39] formalizes the correctness of this approach (we only need to take into account the replacement restrictions indicated by the replacement map μ).

Proposition 2. *Let $(\mathcal{R},\mu)$ be a CS-TRS. If there is an arc from $u \to v$ to $u' \to v'$ in the ICS-dependency graph, then v and u' are innermost μ-connectable.*

Example 6. (Continuing Example 2) Since $\text{REN}_u^{\mu^\sharp}(\text{CAP}_u^{\mu^\sharp}(\texttt{F(X,X)})) = \texttt{F(X,X)}$ and `F(c(Y),Y)` do not unify we conclude (and this can easily be implemented) that the ICS-dependency graph for the CS-TRS $(\mathcal{R},\mu)$ in Example 2 contains no cycles.

We know how to approximate the ICS-dependency graph by means of the functions CAP_u^μ and REN_u^μ. The next step is checking the innermost μ-termination with the ICSDG automatically.

4.2 Proofs of Termination of Innermost *CSR* Using the ICSDG

The absence of infinite innermost $(\mathcal{R},\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp)$-chains is checked in the ICSDG by finding (possibly different) μ-reduction pairs $(\gtrsim_\mathfrak{C},\sqsupset_\mathfrak{C})$ for each cycle $\mathfrak{C}$. Here, a μ-reduction pair is a pair $(\gtrsim,\sqsupset)$ where $\gtrsim$ is a stable and μ-monotonic quasi-ordering which is compatible with the well-founded and stable ordering $\sqsupset$, i.e., satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$.

Theorem 2 (Use of the ICSDG). *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. Then, $\mathcal{R}$ is innermost μ-terminating iff for each cycle $\mathfrak{C}$ in the innermost context-sensitive dependency graph there is a μ-reduction pair $(\gtrsim_\mathfrak{C},\sqsupset_\mathfrak{C})$ such that $\mathcal{R} \subseteq \gtrsim_\mathfrak{C}$, $\mathfrak{C} \subseteq \gtrsim_\mathfrak{C} \cup \sqsupset_\mathfrak{C}$, and*

1. *If $\mathfrak{C} \cap \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) = \varnothing$, then $\mathfrak{C} \cap \sqsupset_\mathfrak{C} \neq \varnothing$*
2. *If $\mathfrak{C} \cap \mathsf{DP}_\mathcal{X}(\mathcal{R},\mu) \neq \varnothing$, then $\trianglerighteq_\mu \subseteq \gtrsim_\mathfrak{C}$, and*
   (a) $\mathfrak{C} \cap \sqsupset_\mathfrak{C} \neq \varnothing$ and $f(x_1,\ldots,x_k) \gtrsim_\mathfrak{C} f^\sharp(x_1,\ldots,x_k)$ for all $f^\sharp$ in $\mathfrak{C}$, or
   (b) $f(x_1,\ldots,x_k) \sqsupset_\mathfrak{C} f^\sharp(x_1,\ldots,x_k)$ for all $f^\sharp$ in $\mathfrak{C}$.

The proof is similar to that of [1, Theorem 4]. The practical use of Theorem 2 concerns the so-called *strongly connected components* (SCCs) of the dependency graph, rather than the cycles themselves (which are exponentially many) [13,14].

Example 7. There are many examples that are easily solved when trying to build the ICS-dependency graph since they do not contain cycles. This is the case for Example 2 and Example 5.

The use of *argument filterings*, which is standard in the current formulations of the dependency pairs method, also adapts without changes to this setting.

Also the *subterm criterion* [13], can be used to ignore certain cycles of the dependency graph. In [1], we have adapted it to *CSR*.

5 Usable CS-Rules

An interesting feature in the treatment of innermost termination problems using the dependency pairs approach is that, since the variables in the right-hand side of the dependency pairs are in normal form, the rules which can be used to connect contiguous dependency pairs are usually a proper subset of the rules in the TRS. This leads to the notion of *usable rules* [3, Definition 32] which simplifies the proofs of innermost termination of rewriting. We adapt this notion to the context-sensitive setting.

Definition 5 (Basic usable CS-rules). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For any symbol f let $Rules(\mathcal{R}, f)$ be the set of rules defining f and such that the left-hand side l has no redex as proper μ-replacing subterm. For any term t the set of* basic *usable rules $\mathbf{U}_0(\mathcal{R}, t)$ is as follows:*

$$\begin{aligned}\mathbf{U}_0(\mathcal{R}, x) &= \varnothing\\ \mathbf{U}_0(\mathcal{R}, f(t_1, \ldots, t_n)) &= Rules(\mathcal{R}, f) \cup \bigcup_{i \in \mu(f)} \mathbf{U}_0(\mathcal{R}', t_i) \cup \bigcup_{l \to r \in Rules(\mathcal{R}, f)} \mathbf{U}_0(\mathcal{R}', r)\end{aligned}$$

where $\mathcal{R}' = \mathcal{R} - Rules(\mathcal{R}, f)$. If $\mathfrak{C} \subseteq \mathsf{DP}(\mathcal{R}, \mu)$, then $\mathbf{U}_0(\mathcal{R}, \mathfrak{C}) = \bigcup_{l \to r \in \mathfrak{C}} \mathbf{U}_0(\mathcal{R}, r)$.

Interestingly, although our definition is a straightforward extension of the classical one (which just takes into account that μ-rewritings are possible only on μ-replacing subterms), some subtleties arise due to the presence of *non-conservative* rules. Here, a rule $l \to r$ of a TRS $\mathcal{R}$ is μ-conservative if $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$, i.e., it does not contain migrating variables; $\mathcal{R}$ is μ-conservative if all its rules are (see [20]).

Definition 6 (Conservative CSDPs). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. The set of* conservative *CSDPs $\mathsf{DP}_{\mathsf{Co}}(\mathcal{R}, \mu)$ is $\mathsf{DP}_{\mathsf{Co}}(\mathcal{R}, \mu) = \{u \to v \in \mathsf{DP}(\mathcal{R}, \mu) \mid \mathcal{V}ar^{\mu^\sharp}(v) \subseteq \mathcal{V}ar^{\mu^\sharp}(u)\}$.*

Note that $\mathsf{DP}_{\mathsf{Co}}(\mathcal{R}, \mu) \subseteq \mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$. Basic usable rules in Definition 5 can be applied to cycles $\mathfrak{C}$ consisting of *conservative* CS-dependency pairs provided that $\mathbf{U}_0(\mathcal{R}, \mathfrak{C})$ is also conservative. This is proved in Theorem 3 below. First, we need some auxiliary results.

Proposition 3. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $s \overset{i}{\hookrightarrow} s'$ by applying a rule $l \to r \in \mathcal{R}$, then there is a substitution σ' such that $s' = \sigma'(t')$ for $t' = t[r]_p$ and $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$.*

Proof. Let $p \in \mathcal{P}os^{\mu}(s)$ be the position of an innermost redex $s|_p = \theta(l)$ for some substitution θ. Since $s = \sigma(t)$ and for all replacing variables in t, we have $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$, it follows that p is a non-variable (replacing) position of t. Therefore, $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Since $s = \sigma(t)$, we have that $s' = \sigma(t)[\theta(r)]_p$ and since $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$, by defining $\sigma'(x) = \sigma(x)$ for all $x \in Var(t)$ and $\sigma(x) = \theta(x)$ for all $x \in \mathcal{V}ar(r)$ (as usual, we assume $\mathcal{V}ar(t) \cap \mathcal{V}ar(r) = \varnothing$), we have $s' = \sigma'(t[r]_p)$.

Proposition 4. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $s \overset{i}{\hookrightarrow} s'$ by applying a conservative rule $l \to r \in \mathcal{R}$, then there is a substitution σ' such that $s' = \sigma'(t')$ for $t' = t[r]_p$, $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$.*

Proof. By Proposition 3, we know that σ', as in Proposition 3, satisfies $s' = \sigma'(t')$ for θ as in Proposition 3 and some $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Since $s|_p$ is an innermost μ-replacing redex, we have that $\forall y \in \mathcal{V}ar^{\mu}(l)$, $\theta(y) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since the rule $l \to r$ is conservative, $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$, hence $\forall z \in \mathcal{V}ar^{\mu}(r)$, $\sigma'(z) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since $\mathcal{V}ar^{\mu}(t[r]_p) \subseteq \mathcal{V}ar^{\mu}(t) \cup \mathcal{V}ar^{\mu}(r)$, we have that $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$.

Proposition 5. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $\mathbf{U}_0(\mathcal{R}, t)$ is conservative and $s \overset{i}{\hookrightarrow}^*_{\mathcal{R}} u$ then $s \overset{i}{\hookrightarrow}^*_{\mathbf{U}_0(\mathcal{R},t)} u$.*

Proof. By induction on the length of the sequence $s \overset{i}{\hookrightarrow}^*_{\mathcal{R}} u$. If $s = \sigma(t) = u$, it is trivial. Otherwise, if $s \overset{i}{\hookrightarrow}_{\mathcal{R}} s' \overset{i}{\hookrightarrow}^*_{\mathcal{R}} u$, we first prove that the result also holds in $s \overset{i}{\hookrightarrow}_{\mathcal{R}} s'$. By Proposition 3, $s = \sigma(t)$, and $s' = \sigma'(t')$ for $t' = t[r]_p$ is such that $s|_p = \theta(l)$ and $s'|_p = \theta(r)$ for some $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Thus, $root(l) = root(t|_p)$ and by Definition 5, we can conclude that $l \to r \in \mathbf{U}_0(\mathcal{R}, t)$. By hypothesis, $\mathbf{U}_0(\mathcal{R}, t)$ is conservative. Thus, $l \to r$ is conservative and by Proposition 4, $s' = \sigma'(t')$ and $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since $t' = t[r]_p$ and $root(t|_p) = root(l)$, we have that $\mathbf{U}_0(\mathcal{R}, t') \subseteq \mathbf{U}_0(\mathcal{R}, t)$ and (since $\mathbf{U}_0(\mathcal{R}, t)$ is conservative) $\mathbf{U}_0(\mathcal{R}, t')$ is conservative as well. By the induction hypothesis we know that $s' \overset{i}{\hookrightarrow}^*_{\mathbf{U}_0(\mathcal{R},t')} u$. Thus we have $s \overset{i}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},t)} s' \overset{i}{\hookrightarrow}^*_{\mathbf{U}_0(\mathcal{R},t)} u$ as desired.

Theorem 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $\mathfrak{C} \subseteq \mathsf{DP}_{\mathsf{Co}}(\mathcal{R}, \mu)$. If there is a $\mu^\sharp$-reduction pair $(\gtrsim, \sqsupset)$ such that, $\mathbf{U}_0(\mathcal{R}, \mathfrak{C})$ is conservative, $\mathbf{U}_0(\mathcal{R}, \mathfrak{C}) \subseteq \gtrsim$, $\mathfrak{C} \subseteq \gtrsim \cup \sqsupset$, and $\mathfrak{C} \cap \sqsupset \neq \varnothing$, then there is no minimal innermost $(\mathcal{R}, \mathfrak{C}, \mu^\sharp)$-chain.*

Proof. We proceed by contradiction. If $\mathcal{R}$ is not innermost μ-terminating, then by Theorem 1 there is an infinite innermost $(\mathcal{R}, \mathsf{DP}(\mathcal{R}, \mu), \mu^\sharp)$-chain:

$$\sigma(u_1) \hookrightarrow_{\mathsf{DP}(\mathfrak{C},\mu),\mu^\sharp} \sigma(v_1) \overset{i}{\hookrightarrow}^!_{\mathcal{R}} \sigma(u_2) \hookrightarrow_{\mathsf{DP}(\mathfrak{C},\mu),\mu^\sharp} \sigma(v_2) \overset{i}{\hookrightarrow}^!_{\mathcal{R}} \sigma(u_3) \hookrightarrow_{\mathsf{DP}(\mathfrak{C},\mu),\mu^\sharp} \cdots$$

for a substitution σ and $u_i \to v_i \in \mathsf{DP}_{\mathcal{F}}(\mathfrak{C}, \mu)$ for $i \geq 1$. Since $u_i \to v_i \in \mathsf{DP}_{\mathsf{Co}}(\mathcal{R}, \mu)$, and $\sigma(u_i) \in \mathsf{NF}_{\mu}(\mathcal{R})$, this implies that $\forall x \in \mathcal{V}ar^{\mu}(v_i)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$ and by Proposition 5 the sequence can be seen as:

$$\sigma(u_1) \hookrightarrow_{\mathsf{DP}(\mathfrak{C},\mu),\mu^\sharp} \sigma(v_1) \overset{i}{\hookrightarrow}^!_{(\mathbf{U}_0(\mathcal{R},\mathfrak{C}),\mu^\sharp)} \sigma(u_2) \hookrightarrow_{\mathsf{DP}(\mathfrak{C},\mu),\mu^\sharp} \sigma(v_2) \overset{i}{\hookrightarrow}^!_{(\mathbf{U}_0(\mathcal{R},\mathfrak{C}),\mu^\sharp)} \sigma(u_3) \hookrightarrow \cdots$$

By stability of $\sqsupset$, we have $\sigma(u_i) \sqsupset \sigma(v_i)$ and by stability, μ-monotonicity and transitivity of $\gtrsim$ we have that $\sigma(v_i) \gtrsim \sigma(u_{i+1})$. By using the compatibility conditions of the μ-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence which contradicts well-foundedness of $\sqsupset$.

Unfortunately, dealing with non-conservative CSDPs, considering the basic usable CS-rules does *not* ensure a correct approach.

Example 8. Consider again the TRS $\mathcal{R}$:

```
b -> c(b)
f(c(X),X)-> f(X,X)
```

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{c}) = \varnothing$. There are *two* non-conservative CS-DPs (note that $\mu^\sharp(\mathtt{F}) = \mu(\mathtt{f}) = \{1\}$):

```
F(c(X),X) -> F(X,X)
F(c(X),X) -> X
```

and only one cycle in the ICSDG:

```
F(c(X),X) -> F(X,X)
```

Note that $\mathbf{U}_0(\mathcal{R}, \mathtt{F(X,X)}) = \varnothing$. Since this CS-DP is strictly compatible with, e.g., an LPO, we would conclude the innermost μ-termination of $\mathcal{R}$. However, this system is *not* innermost μ-terminating:

$$\mathtt{f(}\underline{\mathtt{b}}\mathtt{,b)} \overset{i}{\hookrightarrow} \underline{\mathtt{f(c(b),b)}} \overset{i}{\hookrightarrow} \mathtt{f(}\underline{\mathtt{b}}\mathtt{,b)} \overset{i}{\hookrightarrow} \cdots$$

The problem is that we have to take into account the special status of variables in the right-hand side of a non-conservative CS-DP. Instances of such variables are *not* guaranteed to be μ-normal forms. For this reason, when a cycle contains at least one non-conservative CS-DP, we have to consider the whole set of rules of the system.

Furthermore, conservativeness of $\mathbf{U}_0(\mathcal{R}, \mathfrak{C})$ cannot be dropped either since we could infer an incorrect result as shown by the following example.

Example 9. Consider the TRS $\mathcal{R}$:

```
b -> c(b)
f(c(X),X) -> f(g(X),X)
g(X) -> X
```

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{g}) = \mu(\mathtt{c}) = \varnothing$. There is only one conservative cycle: {`F(c(X),X) -> F(g(X),X)`} having only one usable (but non-conservative!) rule `g(X) -> X`. This is compatible with the μ-reduction pair induced by the following polynomial interpretation:

$$[\mathtt{f}](x,y) = 0 \qquad [\mathtt{c}](x) = x + 1 \qquad [\mathtt{g}](x) = x \qquad [\mathtt{F}](x,y) = x$$

However the system is not innermost μ-terminating:

$$\underline{\mathtt{f(c(b),b)}} \overset{i}{\hookrightarrow} \mathtt{f(}\underline{\mathtt{g(b)}}\mathtt{,b)} \overset{i}{\hookrightarrow} \mathtt{f(}\underline{\mathtt{b}}\mathtt{,b)} \overset{i}{\hookrightarrow} \underline{\mathtt{f(c(b),b)}} \overset{i}{\hookrightarrow} \cdots$$

Nevertheless, Theorem 3 is useful to improve the proofs of termination of innermost *CSR* as the following example shows.

Example 10. Consider again the TRS $\mathcal{R}$ in example 1. The system contains three cycles in the ICSDG:

```
{ SEL(s(N),cons(X,XS)) -> SEL(N,XS) }
{ MINUS(s(X),s(Y)) -> MINUS(X,Y) }
{ QUOT(s(X),s(Y)) -> QUOT(minus(X,Y),s(Y)) }
```

The first two cycles can be solved by using the subterm criterion. However, without the notion of usable rules, the last one is difficult to solve. The cycle is conservative and the obtained usable rules are also conservative: `minus(X,0) -> X` and `minus(s(X),s(Y)) -> minus(X,Y)`. According to Theorem 3, the cycle can be easily solved by using a polynomial interpretation:

$$[\texttt{minus}](x,y) = x \qquad [\texttt{0}] = 0$$
$$[\texttt{s}](x) = x + 1 \qquad [\texttt{QUOT}](x,y) = x$$

6 Experiments

We have implemented the techniques described in the previous sections as part of the tool MU-TERM [19]. In order to evaluate the techniques which are reported in this paper we have made some benchmarks. We have considered the examples in the Termination Problem Data Base (TPDB, version 3.2) available through the URL:

`http://www.lri.fr/~marche/tpdb/`

Although there is no special TPDB category for innermost termination of *CSR* (yet) we have used the TRS/CSR directory in order to test our techniques for proving termination of innermost *CSR* (Theorems 2, 3). It contains 90 examples of CS-TRSs. We are able to give an automatic proof of innermost μ-termination for 62 examples. In order to evaluate our direct techniques in comparison with the transformational approach of [7,8,16], where termination of innermost *CSR* for a CS-TRS $(\mathcal{R}, \mu)$ is proved by proving innermost termination of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$, where Θ specifies a particular transformation (see [6,7] for a survey on this topic), we have transformed the set of examples by using the transformations that are correct for proving innermost termination of *CSR*: Giesl and Middeldorp's correct transformations for proving termination of innermost *CSR*, see [7], although we use the 'authors-based' notation introduced in [20]: GM and C for transformations 1 and 2 for proving termination of *CSR* introduced in [8], and iGM for the specific transformation for proving termination of innermost *CSR* introduced in [7]. Then we have proved innermost termination of the set of examples with AProVE [9], which is able to prove innermost termination of standard rewriting. In fact, AProVE is currently the most powerful tool for proving termination and innermost termination of TRSs but as we have said, MU-TERM is nowadays the only termination tool that proves innermost termination of *CSR*. The results are summarized in Table 1. Further details can be found here:

`http://www.dsic.upv.es/~balarcon/FroCoS07/benchmarks`

Indirectly, we have also made the first benchmarks to evaluate the existing correct transformations for proving innermost termination of *CSR* (see Table 1)

Table 1. Comparing techniques for proving termination of innermost *CSR*

	iCSDPs	Transformations
YES score	62	44
YES average time	0.13 sec.	5 sec.

	C	GM	iGM
YES score	24	41	30

showing that, quite surprisingly, the iGM transformation (which is in principle the more suitable one for proving innermost termination of *CSR*) obtains worse results than GM.

7 Conclusions and Future Work

In this paper, we have extended the context-sensitive dependency pairs approach in [1] for proving termination of innermost *CSR*. We have introduced the notion of an innermost μ-chain (Definition 2) and proved that it can be used to characterize innermost μ-termination (Theorem 1). We have also shown how to automatically prove innermost μ-termination by means of the ICS-dependency graph (Definition 4, Theorem 2). We have formulated the notion of basic usable rules showing how to use them in proofs of innermost termination of *CSR* (Definition 5, Theorem 3). We have implemented these techniques in MU-TERM and have made some benchmarks.

Up to now, no direct method has been proposed to prove termination of innermost *CSR*. So this is the first proposal of a direct method for proving termination of innermost *CSR*. We have extended Arts and Giesl's approach to prove innermost termination of TRSs to *CSR* (thus also extending [1,2]). The main issue which is left open is a general notion of *usable rules*, which can be used with non-conservative CSDPs. As in the standard case, this would probably help us to achieve better results. Even without them, though, our benchmarks show that the use of CSDPs dramatically improves the performance of existing (transformational) methods for proving termination of innermost *CSR*.

Acknoledgements. We thank the anonymous referees for many useful remarks.

References

1. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 297–308. Springer, Heidelberg (2006)
2. Alarcón, B., Gutiérrez, R., Lucas, S.: Improving the context-sensitive dependency graph. Electronic Notes in Theoretical Computer Science (2007) (to appear)
3. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236, 133–178 (2000)
4. Borralleras, C.: Ordering-based methods for proving termination automatically. PhD Thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya (May 2003)

5. Giesl, J., Arts, T., Ohlebusch, E.: Modular Termination Proofs for Rewriting Using Dependency Pairs. Journal of Symbolic Computation 34(1), 21–58 (2002)
6. Giesl, J., Middeldorp, A.: Innermost termination of context-sensitive rewriting. Aachener Informatik-Berichte (AIBs) 2002-04, RWTH Aachen (2002)
7. Giesl, J., Middeldorp, A.: Innermost termination of context-sensitive rewriting. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 231–244. Springer, Heidelberg (2003)
8. Giesl, J., Middeldorp, A.: Transformation techniques for context-sensitive rewrite systems. Journal of Functional Programming 14(4), 379–427 (2004)
9. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 281–286. Springer, Heidelberg (2006)
10. Gramlich, B.: Abstract Relations between Restricted Termination and Confluence Properties of Rewrite Systems. Fundamenta Informaticae 24, 3–23 (1995)
11. Gramlich, B.: On Proving Termination by Innermost Termination. In: Ganzinger, H. (ed.) Rewriting Techniques and Applications. LNCS, vol. 1103, pp. 93–107. Springer, Heidelberg (1996)
12. Gramlich, B., Lucas, S.: Modular Termination of Context-Sensitive Rewriting. In: Proc. of PPDP'02, pp. 50–61. ACM Press, New York (2002)
13. Hirokawa, N., Middeldorp, A.: Dependency Pairs Revisited. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 249–268. Springer, Heidelberg (2004)
14. Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. Information and Computation 199, 172–199 (2005)
15. Lucas, S.: Context-sensitive computations in functional and functional logic programs. Journal of Functional and Logic Programming 1998(1), 1–61 (1998)
16. Lucas, S.: Termination of Rewriting With Strategy Annotations. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001. LNCS (LNAI), vol. 2250, pp. 669–684. Springer, Heidelberg (2001)
17. Lucas, S.: Termination of on-demand rewriting and termination of OBJ programs. In: Proc. of PPDP'01, pp. 82–93. ACM Press, New York (2001)
18. Lucas, S.: Context-sensitive rewriting strategies. Information and Computation 178(1), 293–343 (2002)
19. Lucas, S.: MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 200–209. Springer, Heidelberg (2004)
20. Lucas, S.: Proving termination of context-sensitive rewriting by transformation. Information and Computation 204(12), 1782–1846 (2006)
21. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer, Heidelberg (2002)
22. TeReSe (ed.).: Term Rewriting Systems. Cambridge University Press, Cambridge (2003)

18.5 Improving Context-Sensitive Dependency Pairs

5. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider -Kamp, and R. Thiemann. **Improving Context-Sensitive Dependency Pairs**. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of XV International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *Lecture Notes in Computer Science*, pages 636–651. Springer-Verlag, 2008.

Improving Context-Sensitive Dependency Pairs*

Beatriz Alarcón[1], Fabian Emmes[2], Carsten Fuhs[2], Jürgen Giesl[2], Raúl Gutiérrez[1], Salvador Lucas[1], Peter Schneider-Kamp[2], and René Thiemann[3]

[1] DSIC, Universidad Politécnica de Valencia, Spain
[2] LuFG Informatik 2, RWTH Aachen University, Germany
[3] Institute of Computer Science, University of Innsbruck, Austria

Abstract. Context-sensitive dependency pairs (CS-DPs) are currently the most powerful method for automated termination analysis of context-sensitive rewriting. However, compared to DPs for ordinary rewriting, CS-DPs suffer from two main drawbacks: (a) CS-DPs can be *collapsing*. This complicates the handling of CS-DPs and makes them less powerful in practice. (b) There does not exist a "*DP framework*" for CS-DPs which would allow one to apply them in a flexible and modular way. This paper solves drawback (a) by introducing a new definition of CS-DPs. With our definition, CS-DPs are always non-collapsing and thus, they can be handled like ordinary DPs. This allows us to solve drawback (b) as well, i.e., we extend the existing DP framework for ordinary DPs to context-sensitive rewriting. We implemented our results in the tool AProVE and successfully evaluated them on a large collection of examples.

1 Introduction

Context-sensitive rewriting [23,24] models evaluations in programming languages. It uses a *replacement map* μ with $\mu(f) \subseteq \{1, ..., \text{arity}(f)\}$ for every function symbol f to specify the argument positions of f where rewriting may take place.

Example 1. Consider this context-sensitive term rewrite system (CS-TRS)

$$\begin{array}{rlrl}
\mathsf{gt}(0, y) &\to \mathsf{false} & \mathsf{p}(0) &\to 0 \\
\mathsf{gt}(\mathsf{s}(x), 0) &\to \mathsf{true} & \mathsf{p}(\mathsf{s}(x)) &\to x \\
\mathsf{gt}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{gt}(x, y) & \mathsf{minus}(x, y) &\to \mathsf{if}(\mathsf{gt}(y, 0), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x) \quad (1) \\
\mathsf{if}(\mathsf{true}, x, y) &\to x & \mathsf{div}(0, \mathsf{s}(y)) &\to 0 \\
\mathsf{if}(\mathsf{false}, x, y) &\to y & \mathsf{div}(\mathsf{s}(x), \mathsf{s}(y)) &\to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x, y), \mathsf{s}(y)))
\end{array}$$

with $\mu(\mathsf{if}) = \{1\}$ and $\mu(f) = \{1, \ldots, \text{arity}(f)\}$ for all other symbols f to model the usual behavior of if*: in $\mathsf{if}(t_1, t_2, t_3)$, one may evaluate t_1, but not t_2 or t_3. It will turn out that due to μ, this CS-TRS is indeed terminating. In contrast, if one allows arbitrary reductions, then the TRS would benon-terminating:*

* Authors from Valencia were partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grants TIN 2007-68093-C02-02 and HA 2006-0007. B. Alarcón was partially supported by the Spanish MEC/MICINN under FPU grant AP2005-3399. R. Gutiérrez was partially supported by the Spanish MEC/MICINN, under grant TIN 2004-7943-C04-02. Authors from Aachen were supported by the DAAD under grant D/06/12785 and by the DFG under grant GI 274/5-2.

I. Cervesato, H. Veith, and A. Voronkov (Eds.): LPAR 2008, LNCS 5330, pp. 636–651, 2008.
© Springer-Verlag Berlin Heidelberg 2008

$\mathsf{minus}(0,0) \to^+ \mathsf{if}(\mathsf{gt}(0,0),\mathsf{minus}(0,0),0) \to^+ \mathsf{if}(...,\mathsf{if}(\mathsf{gt}(0,0),\mathsf{minus}(0,0),0),...) \to^+ ...$

There are two approaches to prove termination of context-sensitive rewriting. The first approach transforms CS-TRSs to ordinary TRSs, cf. [13,26]. But transformations often generate complicated TRSs where all termination tools fail.

Therefore, it is more promising to adapt existing termination techniques from ordinary term rewriting to the context-sensitive setting. Such adaptions were done for classical methods like RPO or polynomial orders [8,19,25]. However, much more powerful techniques like the *dependency pair* (DP) method [6] are implemented in almost all current termination tools for TRSs. But for a long time, it was not clear how to adapt the DP method to context-sensitive rewriting.

This was solved first in [1]. The corresponding implementation in the tool MU-TERM [3] outperformed all previous tools for termination of CS rewriting.

Nevertheless, the existing results on CS-DPs [1,2,4,20] still have major disadvantages compared to the DP method for ordinary rewriting, since CS-DPs can be *collapsing*. To handle such DPs, one has to impose strong requirements which make the CS-DP method quite weak and which make it difficult to extend refined termination techniques based on DPs to the CS case. In particular, the *DP framework* [14,17,21], which is the most powerful formulation of the DP method for ordinary TRSs, has not yet been adapted to the CS setting.

In this paper, we solve these problems. After presenting preliminaries in Sect. 2, we introduce a new notion of *non-collapsing* CS-DPs in Sect. 3. This new notion makes it much easier to adapt termination techniques based on DPs to context-sensitive rewriting. Therefore, Sect. 4 extends the *DP framework* to the context-sensitive setting and shows that existing methods from this framework only need minor changes to apply them to context-sensitive rewriting.

All our results are implemented in the termination prover AProVE [16]. As shown by the empirical evaluation in Sect. 5, our contributions improve the power of automated termination analysis for context-sensitive rewriting substantially.

2 Context-Sensitive Rewriting and CS-Dependency Pairs

See [7] and [23] for basics on term rewriting and context-sensitive rewriting, respectively. Let $\mathcal{P}os(s)$ be the set of *positions* of a term s. For a replacement map μ, we define the *active positions* $\mathcal{P}os^\mu(s)$: For $x \in \mathcal{V}$ let $\mathcal{P}os^\mu(x) = \{\varepsilon\}$ where ε is the root position. Moreover, $\mathcal{P}os^\mu(f(s_1,\ldots,s_n)) = \{\varepsilon\} \cup \{i\,p \mid i \in \mu(f),\ p \in \mathcal{P}os^\mu(s_i)\}$. We say that $s \trianglerighteq_\mu t$ holds if $t = s|_p$ for some $p \in \mathcal{P}os^\mu(s)$ and $s \triangleright_\mu t$ if $s \trianglerighteq_\mu t$ and $s \neq t$. Moreover, $s \triangleright_\mu t$ if $t = s|_p$ for some $p \in \mathcal{P}os(s) \setminus \mathcal{P}os^\mu(s)$. We denote the ordinary subterm relations by $\trianglerighteq$ and $\triangleright$.

A *CS-TRS* $(\mathcal{R},\mu)$ consists of a finite TRS $\mathcal{R}$ and a replacement map μ. We have $s \hookrightarrow_{\mathcal{R},\mu} t$ iff there are $\ell \to r \in \mathcal{R}$, $p \in \mathcal{P}os^\mu(s)$, and a substitution σ with $s|_p = \sigma(\ell)$ and $t = s[\sigma(r)]_p$. This reduction is an *innermost* step (denoted $\overset{i}{\hookrightarrow}_{\mathcal{R},\mu}$) if all t with $s|_p \triangleright_\mu t$ are in normal form w.r.t. $(\mathcal{R},\mu)$. A term s is in *normal form* w.r.t. $(\mathcal{R},\mu)$ if there is no term t with $s \hookrightarrow_{\mathcal{R},\mu} t$. A CS-TRS $(\mathcal{R},\mu)$ is *terminating* if $\hookrightarrow_{\mathcal{R},\mu}$ is well founded and *innermost terminating* if $\overset{i}{\hookrightarrow}_{\mathcal{R},\mu}$ is well founded.

Let $\mathcal{D} = \{root(\ell) \mid \ell \to r \in \mathcal{R}\}$ be the set of *defined symbols*. For every $f \in \mathcal{D}$, let $f^\sharp$ be a fresh *tuple symbol* of same arity, where we often write "F" instead of "$f^\sharp$". For $t = f(t_1, \ldots, t_n)$ with $f \in \mathcal{D}$, let $t^\sharp = f^\sharp(t_1, \ldots, t_n)$.

Definition 2 (CS-DPs [1]). *Let $(\mathcal{R}, \mu)$ be a CS-TRS. If $\ell \to r \in \mathcal{R}$, $r \trianglerighteq_\mu t$, and* root$(t) \in \mathcal{D}$, *then $\ell^\sharp \to t^\sharp$ is an* ordinary dependency pair.[1] *If $\ell \to r \in \mathcal{R}$, $r \trianglerighteq_\mu x$ for a variable x, and $\ell \not\rhd_\mu x$, then $\ell^\sharp \to x$ is a* collapsing DP. *Let $\mathsf{DP}_o(\mathcal{R}, \mu)$ and $\mathsf{DP}_c(\mathcal{R}, \mu)$ be the sets of all ordinary resp. all collapsing DPs.*

Example 3. For the TRS of Ex. 1, we obtain the following CS-DPs.

$$\mathsf{GT}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{GT}(x, y) \quad (2)$$
$$\mathsf{IF}(\mathsf{true}, x, y) \to x \quad (3)$$
$$\mathsf{IF}(\mathsf{false}, x, y) \to y \quad (4)$$
$$\mathsf{M}(x, y) \to \mathsf{IF}(\mathsf{gt}(y, \mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x) \quad (5)$$
$$\mathsf{M}(x, y) \to \mathsf{GT}(y, \mathsf{0}) \quad (6)$$
$$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x, y), \mathsf{s}(y)) \quad (7)$$
$$\mathsf{D}(\mathsf{s}(x), \mathsf{s}(y)) \to \mathsf{M}(x, y) \quad (8)$$

To prove termination, one has to show that there is no infinite *chain* of DPs. For ordinary rewriting, a sequence $s_1 \to t_1, s_2 \to t_2, \ldots$ of DPs is a *chain* if there is a substitution σ such that $t_i\sigma$ reduces to $s_{i+1}\sigma$.[2] If all $t_i\sigma$ are terminating, then the chain is *minimal* [14,17,22]. But due to the collapsing DPs, the notion of "chains" has to be adapted when it is used with CS-DPs [1]. If $s_i \to t_i$ is a collapsing DP (i.e., if $t_i \in \mathcal{V}$), then instead of $t_i\sigma \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}\sigma$ (and termination of $t_i\sigma$ for minimality), one requires that there is a term w_i with $t_i\sigma \trianglerighteq_\mu w_i$ and $w_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}\sigma$. For minimal chains, $w_i^\sharp$ must be terminating.

Example 4. Ex. 1 has the chain (5), (3), (5) as $\mathsf{IF}(\mathsf{gt}(\mathsf{s}(y), \mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y))), x) \hookrightarrow^*_{\mathcal{R},\mu} \mathsf{IF}(\mathsf{true}, \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y))), x) \hookrightarrow_{(3),\mu} \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y)))$ *and* $(\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y))))^\sharp = \mathsf{M}(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y)))$ *is an instance of the left-hand side of (5).*

A CS-TRS is terminating iff there is no infinite chain [1]. As in the non-CS case, the above notion of chains can also be adapted to *innermost* rewriting. Then a CS-TRS is innermost terminating iff there is no infinite innermost chain [4].

Due to the collapsing CS-DPs (and the corresponding definition of "chains"), it is not easy to extend existing techniques for proving absence of infinite chains to CS-DPs. Therefore, we now introduce a new improved definition of CS-DPs.

3 Non-collapsing CS-Dependency Pairs

Ordinary DPs only consider active subterms of right-hand sides. So Rule (1) of Ex. 1 only leads to the DP (5), but not to $\mathsf{M}(x, y) \to \mathsf{M}(\mathsf{p}(x), \mathsf{p}(y))$. However, the inactive subterm $\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))$ of the right-hand side of (1) may become active again when applying the rule $\mathsf{if}(\mathsf{true}, x, y) \to x$. Therefore, Def. 2 creates a collapsing DP like (3) whenever a rule $\ell \to r$ has a *migrating variable* x with $r \trianglerighteq_\mu x$, but $\ell \not\rhd_\mu x$. Indeed, when instantiating the *collapse-variable* x in (3) with an instance of the "hidden term" $\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))$, one obtains a chain which simulates the rewrite sequence from $\mathsf{minus}(t_1, t_2)$ over $\mathsf{if}(..., \mathsf{minus}(\mathsf{p}(t_1), \mathsf{p}(t_2)), ...)$

[1] A refinement is to eliminate DPs where $\ell \rhd_\mu t$, cf. [1,9].

[2] We always assume that different occurrences of DPs are variable-disjoint and consider substitutions whose domains may be infinite.

to $\mathsf{minus}(\mathsf{p}(t_1), \mathsf{p}(t_2))$, cf. Ex. 4. Our main observation is that collapsing DPs are only needed for certain instantiations of the variables. One might be tempted to allow only instantiations of collapse-variables by *hidden terms*.[3]

Definition 5 (Hidden Term). *Let $(\mathcal{R}, \mu)$ be a CS-TRS. We say that t is a* hidden term *if $root(t) \in \mathcal{D}$ and if there exists a rule $\ell \to r \in \mathcal{R}$ with $r \rhd_\mu t$.*

In Ex. 1, the only hidden term is $\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))$. But unfortunately, only allowing instantiations of collapse-variables with hidden terms would be unsound.

Example 6. Consider $\mu(\mathsf{g}) = \{1\}$, $\mu(\mathsf{a}) = \mu(\mathsf{b}) = \mu(\mathsf{f}) = \mu(\mathsf{h}) = \varnothing$ and the rules

$$\begin{array}{rcl} \mathsf{a} \to \mathsf{f}(\mathsf{g}(\mathsf{b})) & (9) & \mathsf{h}(x) \to x \\ \mathsf{f}(x) \to \mathsf{h}(x) & & \mathsf{b} \to \mathsf{a} \end{array}$$

The CS-TRS has the following infinite rewrite sequence:

$$\mathsf{a} \hookrightarrow_{\mathcal{R},\mu} \mathsf{f}(\mathsf{g}(\mathsf{b})) \hookrightarrow_{\mathcal{R},\mu} \underline{\mathsf{h}(\mathsf{g}(\mathsf{b})) \hookrightarrow_{\mathcal{R},\mu} \mathsf{g}(\mathsf{b})} \hookrightarrow_{\mathcal{R},\mu} \mathsf{g}(\mathsf{a}) \hookrightarrow_{\mathcal{R},\mu} \dots$$

We obtain the following CS-DPs according to Def. 2:

$$\begin{array}{rcl} \mathsf{A} \to \mathsf{F}(\mathsf{g}(\mathsf{b})) & & \mathsf{H}(x) \to x \\ \mathsf{F}(x) \to \mathsf{H}(x) & & \mathsf{B} \to \mathsf{A} \end{array} \qquad (10)$$

The only hidden term is b, obtained from Rule (9). There is also an infinite chain that corresponds to the infinite reduction above. However, here the collapse-variable x in the DP (10) must be instantiated by $\mathsf{g}(\mathsf{b})$ and not by the hidden term b, cf. the underlined part above. So if one replaced (10) by $\mathsf{H}(\mathsf{b}) \to \mathsf{b}$, there would be no infinite chain anymore and one would falsely conclude termination.

The problem in Ex. 6 is that rewrite rules may add additional symbols like g *above* hidden terms. This can happen if a term $\mathsf{g}(t)$ occurs at an inactive position in a right-hand side and if an instantiation of t could possibly reduce to a term containing a hidden term (i.e., if t has a defined symbol or a variable at an active position). Then we call $\mathsf{g}(\Box)$ a *hiding context*, since it can "hide" a hidden term. Moreover, the composition of hiding contexts is again a hiding context.

Definition 7 (Hiding Context). *Let $(\mathcal{R}, \mu)$ be a CS-TRS. The function symbol f* hides *position i if there is a rule $\ell \to r \in \mathcal{R}$ with $r \rhd_\mu f(r_1, \dots, r_i, \dots, r_n)$, $i \in \mu(f)$, and r_i contains a defined symbol or a variable at an active position. A context C is* hiding *iff $C = \Box$ or C has the form $f(t_1, \dots, t_{i-1}, C', t_{i+1}, \dots, t_n)$ where f hides position i and C' is a hiding context.*

Example 8. In Ex. 6, g hides position 1 due to Rule (9). So the hiding contexts are $\Box, \mathsf{g}(\Box), \mathsf{g}(\mathsf{g}(\Box)), \dots$ In the TRS of Ex. 1, minus hides both positions 1 and 2 and p hides position 1 due to Rule (1). So the hiding contexts are $\Box, \mathsf{p}(\Box), \mathsf{minus}(\Box, \Box), \mathsf{p}(\mathsf{p}(\Box)), \mathsf{minus}(\Box, \mathsf{p}(\Box)), \dots$

To remove collapsing DPs $s \to x$, we now restrict ourselves to instantiations of x with terms of the form $C[t]$ where C is a hiding context and t is a hidden term. So in Ex. 6, the variable x in the DP (10) should only be instantiated by b, $\mathsf{g}(\mathsf{b})$,

[3] A similar notion of *hidden symbols* was presented in [2,4], but there one only used these symbols to improve one special termination technique (the *dependency graph*).

$\mathsf{g}(\mathsf{g}(\mathsf{b}))$, etc. To represent these infinitely many instantiations in a finite way, we replace $s \to x$ by new *unhiding* DPs (which "unhide" hidden terms).

Definition 9 (Improved CS-DPs). *For a CS-TRS* $(\mathcal{R}, \mu)$*, if* $\mathsf{DP}_c(\mathcal{R}, \mu) \neq \varnothing$*, we introduce a fresh*[4] unhiding tuple symbol U *and the following* unhiding DPs*:*

- $s \to \mathsf{U}(x)$ *for every* $s \to x \in \mathsf{DP}_c(\mathcal{R}, \mu)$*,*
- $\mathsf{U}(f(x_1, \ldots, x_i, \ldots, x_n)) \to \mathsf{U}(x_i)$ *for every function symbol* f *of any arity* n *and every* $1 \leq i \leq n$ *where* f *hides position* i*, and*
- $\mathsf{U}(t) \to t^\sharp$ *for every hidden term* t*.*

Let $\mathsf{DP}_u(\mathcal{R}, \mu)$ *be the set of all unhiding DPs (where* $\mathsf{DP}_u(\mathcal{R}, \mu) = \varnothing$*, if* $\mathsf{DP}_c(\mathcal{R}, \mu) = \varnothing$*). Then the set of* improved CS-DPs *is* $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_o(\mathcal{R}, \mu) \cup \mathsf{DP}_u(\mathcal{R}, \mu)$*.*

Example 10. In Ex. 6, instead of (10) we get the unhiding DPs

$$\mathsf{H}(x) \to \mathsf{U}(x), \qquad \mathsf{U}(\mathsf{g}(x)) \to \mathsf{U}(x), \qquad \mathsf{U}(\mathsf{b}) \to \mathsf{B}.$$

Now there is indeed an infinite chain. In Ex. 1, instead of (3) and (4), we obtain:[5]

$$\mathsf{IF}(\mathsf{true}, x, y) \to \mathsf{U}(x) \quad (11)$$
$$\mathsf{IF}(\mathsf{false}, x, y) \to \mathsf{U}(y) \quad (12)$$
$$\mathsf{U}(\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))) \to \mathsf{M}(\mathsf{p}(x), \mathsf{p}(y)) \quad (13)$$
$$\mathsf{U}(\mathsf{p}(x)) \to \mathsf{P}(x) \quad (14)$$
$$\mathsf{U}(\mathsf{p}(x)) \to \mathsf{U}(x) \quad (15)$$
$$\mathsf{U}(\mathsf{minus}(x, y)) \to \mathsf{U}(x) \quad (16)$$
$$\mathsf{U}(\mathsf{minus}(x, y)) \to \mathsf{U}(y) \quad (17)$$

Clearly, the improved CS-DPs are never collapsing. Thus, now the definition of (minimal)[6] *chains* is completely analogous to the one for ordinary rewriting.

Definition 11 (Chain). *Let* $\mathcal{P}$ *and* $\mathcal{R}$ *be TRSs and let* μ *be a replacement map. We extend* μ *to tuple symbols by defining* $\mu(f^\sharp) = \mu(f)$ *for all* $f \in \mathcal{D}$ *and* $\mu(\mathsf{U}) = \varnothing$*.*[7] *A sequence of pairs* $s_1 \to t_1, s_2 \to t_2, \ldots$ *from* $\mathcal{P}$ *is a* $(\mathcal{P}, \mathcal{R}, \mu)$-chain *iff there is a substitution* σ *with* $t_i\sigma \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}\sigma$ *and* $t_i\sigma$ *is terminating w.r.t.* $(\mathcal{R}, \mu)$ *for all* i*. It is an* innermost $(\mathcal{P}, \mathcal{R}, \mu)$-chain *iff* $t_i\sigma \overset{\mathsf{i}}{\hookrightarrow}^*_{\mathcal{R},\mu} s_{i+1}\sigma$*,* $s_i\sigma$ *is in normal form, and* $t_i\sigma$ *is innermost terminating w.r.t.* $(\mathcal{R}, \mu)$ *for all* i*.*

Our main theorem shows that improved CS-DPs are still sound and complete.

Theorem 12 (Soundness and Completeness of Improved CS-DPs). *A CS-TRS* $(\mathcal{R}, \mu)$ *is terminating iff there is no infinite* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu)$*-chain and innermost terminating iff there is no infinite innermost* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu)$*-chain.*

Proof. We only prove the theorem for "full" termination. The proof for innermost termination is very similar and can be found in [5].

[4] Alternatively, one could also use different U-symbols for different collapsing DPs.

[5] We omitted the DP $\mathsf{U}(\mathsf{p}(y)) \to \mathsf{P}(y)$ that is "identical" to (14).

[6] Since we only regard *minimal* chains in the following, we included the "minimality requirement" in Def. 11, i.e., we require that all $t_i\sigma$ are (innermost) terminating. As in the DP framework for ordinary rewriting, this restriction to minimal chains is needed for several DP processors (e.g., for the reduction pair processor of Thm. 21).

[7] We define $\mu(\mathsf{U}) = \varnothing$, since the purpose of U is only to remove context around hidden terms. But during this removal, U's argument should not be evaluated.

Soundness

$\mathcal{M}_{\infty,\mu}$ contains all *minimal non-terminating terms*: $t \in \mathcal{M}_{\infty,\mu}$ iff t is non-terminating and every r with $t \rhd_\mu r$ terminates. A term u has the *hiding property* iff

- $u \in \mathcal{M}_{\infty,\mu}$ and
- whenever $u \rhd\!\!\!\!/_\mu s \trianglerighteq_\mu t'$ for some terms s and t' with $t' \in \mathcal{M}_{\infty,\mu}$, then t' is an instance of a hidden term and $s = C[t']$ for some hiding context C.

We first prove the following claim:

$$\text{Let } u \text{ be a term with the hiding property and let } u \hookrightarrow_{\mathcal{R},\mu} v \trianglerighteq_\mu w \text{ with } w \in \mathcal{M}_{\infty,\mu}. \text{ Then } w \text{ also has the hiding property.} \tag{18}$$

Let $w \rhd\!\!\!\!/_\mu s \trianglerighteq_\mu t'$ for some terms s and t' with $t' \in \mathcal{M}_{\infty,\mu}$. Clearly, this also implies $v \rhd\!\!\!\!/_\mu s$. If already $u \rhd s$, then we must have $u \rhd\!\!\!\!/_\mu s$ due to the minimality of u. Thus, t' is an instance of a hidden term and $s = C[t']$ for a hiding context C, since u has the hiding property. Otherwise, $u \not\rhd s$. There must be a rule $\ell \to r \in \mathcal{R}$, an active context D (i.e., a context where the hole is at an active position), and a substitution δ such that $u = D[\delta(\ell)]$ and $v = D[\delta(r)]$. Clearly, $u \not\rhd s$ implies $\delta(\ell) \not\rhd s$ and $D \not\rhd s$. Hence, $v \rhd\!\!\!\!/_\mu s$ means $\delta(r) \rhd\!\!\!\!/_\mu s$. (The root of s cannot be above $\Box$ in D since those positions would be active.) Note that s cannot be at or below a variable position of r, because this would imply $\delta(\ell) \rhd s$. Thus, s is an instance of a non-variable subterm of r that is at an inactive position. So there is a $r' \notin \mathcal{V}$ with $r \rhd\!\!\!\!/_\mu r'$ and $s = \delta(r')$. Recall that $s \trianglerighteq_\mu t'$, i.e., there is a $p \in \mathcal{P}os^\mu(s)$ with $s|_p = t'$. If p is a non-variable position of r', then $\delta(r'|_p) = t'$ and $r'|_p$ is a subterm with defined root at an active position (since $t' \in \mathcal{M}_{\infty,\mu}$ implies $\mathrm{root}(t') \in \mathcal{D}$). Hence, $r'|_p$ is a hidden term and thus, t' is an instance of a hidden term. Moreover, any instance of the context $C' = r'[\Box]_p$ is hiding. So if we define C to be $\delta(C')$, then $s = \delta(r') = \delta(r')[t']_p = \delta(C')[t'] = C[t']$ for the hiding context C. On the contrary, if p is not a non-variable position of r', then $p = p_1\, p_2$ where $r'|_{p_1}$ is a variable x. Now t' is an active subterm of $\delta(x)$ (more precisely, $\delta(x)|_{p_2} = t'$). Since x also occurs in ℓ, we have $\delta(\ell) \rhd \delta(x)$ and thus $u \rhd \delta(x)$. Due to the minimality of u this implies $u \rhd\!\!\!\!/_\mu \delta(x)$. Since $u \rhd\!\!\!\!/_\mu \delta(x) \trianglerighteq_\mu t'$, the hiding property of u implies that t' is an instance of a hidden term and that $\delta(x) = \overline{C}[t']$ for a hiding context $\overline{C}$. Note that since $r'|_{p_1}$ is a variable, the context C' around this variable is also hiding (i.e., $C' = r'[\Box]_{p_1}$). Thus, the context $C = \delta(C')[\overline{C}]$ is hiding as well and $s = \delta(r') = \delta(r')[\delta(x)[t']_{p_2}]_{p_1} = \delta(C')[\overline{C}[t']] = C[t']$.

Proof of Thm. 12 using Claim (18)

If $\mathcal{R}$ is not terminating, then there is a $t \in \mathcal{M}_{\infty,\mu}$ that is minimal w.r.t. $\trianglerighteq$. So there are t, t_i, s_i, t'_{i+1} such that

$$t \xrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu}\ t_1 \xrightarrow{\varepsilon}_{\mathcal{R}} s_1 \trianglerighteq_\mu t'_2 \xrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu}\ t_2 \xrightarrow{\varepsilon}_{\mathcal{R}} s_2 \trianglerighteq_\mu t'_3 \xrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu}\ t_3 \ldots \tag{19}$$

where $t_i, t'_i \in \mathcal{M}_{\infty,\mu}$ and all proper subterms of t (also at *inactive* positions) terminate. Here, "ε" (resp. "$> \varepsilon$") denotes reductions at (resp. strictly below) the root.

Note that (18) implies that all t_i have the hiding property. To see this, we use induction on i. Since t trivially has the hiding property (as it has no non-terminating proper subterms) and all terms in the reduction $t \xhookrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu} t_1$ are from $\mathcal{M}_{\infty,\mu}$ (as both $t, t_1 \in \mathcal{M}_{\infty,\mu}$), we conclude that t_1 also has the hiding property by applying (18) repeatedly. In the induction step, if t_{i-1} has the hiding property, then one application of (18) shows that t'_i also has the hiding property. By applying (18) repeatedly, one then also shows that t_i has the hiding property.

Now we show that $t_i^\sharp \to^+_{\mathsf{DP}(\mathcal{R},\mu)} {t'_{i+1}}^\sharp$ and that all terms in the reduction $t_i^\sharp \to^+_{\mathsf{DP}(\mathcal{R},\mu)} {t'_{i+1}}^\sharp$ terminate w.r.t. $(\mathcal{R},\mu)$. As ${t'_{i+1}}^\sharp \xhookrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu} t_{i+1}^\sharp$, we get an infinite $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu)$-chain.

From (19) we know that there are $\ell_i \to r_i \in \mathcal{R}$ and $p_i \in \mathcal{P}os^\mu(s_i)$ with $t_i = \ell_i\sigma$, $s_i = r_i\sigma$, and $s_i|_{p_i} = r_i\sigma|_{p_i} = t'_{i+1}$ for all i. First let $p_i \in \mathcal{P}os(r_i)$ with $r_i|_{p_i} \notin \mathcal{V}$. Then $\ell_i^\sharp \to (r_i|_{p_i})^\sharp \in \mathsf{DP}_o(\mathcal{R},\mu)$ and $t_i^\sharp = \ell_i^\sharp\sigma \to_{\mathsf{DP}_o(\mathcal{R},\mu)} (r_i|_{p_i})^\sharp\sigma = {t'_{i+1}}^\sharp$. Moreover, as $t_i, t'_{i+1} \in \mathcal{M}_{\infty,\mu}$, the terms $t_i^\sharp$ and ${t'_{i+1}}^\sharp$ are terminating.

Now let p_i be at or below the position of a variable x_i in r_i. By minimality of t_i, x_i only occurs at inactive positions of ℓ_i. Thus, $\ell_i^\sharp \to \mathsf{U}(x_i) \in \mathsf{DP}_u(\mathcal{R},\mu)$ and $r_i = C_i[x_i]$ where C_i is an active context. Recall that $t_i = \ell_i\sigma$ has the hiding property and that $t_i \rhd_\mu \sigma(x_i) \trianglerighteq_\mu t'_{i+1}$. Thus, we have $\sigma(x_i) = C[t'_{i+1}]$ for a hiding context C and moreover, t'_{i+1} is an instance of a hidden term. Hence we obtain:

$$
\begin{array}{lll}
t_i^\sharp = \sigma(\ell_i^\sharp) & & \\
\to_{\mathsf{DP}_u(\mathcal{R},\mu)} \mathsf{U}(\sigma(x_i)) & \text{since } \ell_i^\sharp \to \mathsf{U}(x_i) \in \mathsf{DP}_u(\mathcal{R},\mu) \\
= \mathsf{U}(C[t'_{i+1}]) & \text{for a hiding context } C \\
\to^*_{\mathsf{DP}_u(\mathcal{R},\mu)} \mathsf{U}(t'_{i+1}) & \text{since } \mathsf{U}(C[x]) \to^*_{\mathsf{DP}_u(\mathcal{R},\mu)} \mathsf{U}(x) \text{ for any hiding context } C \\
\to_{\mathsf{DP}_u(\mathcal{R},\mu)} {t'_{i+1}}^\sharp & \text{since } t'_{i+1} \text{ is an instance of a hidden term and} \\
 & \mathsf{U}(t) \to_{\mathsf{DP}_u(\mathcal{R},\mu)} t^\sharp \text{ for any instance } t \text{ of a hidden term}
\end{array}
$$

All terms in the reduction above are terminating. The reason is that again $t_i, t'_{i+1} \in \mathcal{M}_{\infty,\mu}$ implies that $t_i^\sharp$ and ${t'_{i+1}}^\sharp$ are terminating. Moreover, all terms $\mathsf{U}(\ldots)$ are normal forms since $\mu(\mathsf{U}) = \varnothing$ and since U does not occur in $\mathcal{R}$.

Completeness

Let there be an infinite chain $v_1 \to w_1, v_2 \to w_2, \ldots$ of improved CS-DPs. First, let the chain have an infinite tail consisting only of DPs of the form $\mathsf{U}(f(x_1,\ldots,x_i,\ldots,x_n)) \to \mathsf{U}(x_i)$. Since $\mu(\mathsf{U}) = \varnothing$, there are terms t_i with $\mathsf{U}(t_1) \xrightarrow{\varepsilon}_{\mathsf{DP}(R,\mu)} \mathsf{U}(t_2) \xrightarrow{\varepsilon}_{\mathsf{DP}(R,\mu)} \ldots$ Hence, $t_1 \rhd_\mu t_2 \rhd_\mu \ldots$ which contradicts the well-foundedness of $\rhd_\mu$.

Now we regard the remaining case. Here the chain has infinitely many DPs $v \to w$ with $v = \ell^\sharp$ for a rule $\ell \to r \in \mathcal{R}$. Let $v_i \to w_i$ be such a DP and let $v_j \to w_j$ with $j > i$ be the *next* such DP in the chain. Let σ be the substitution used for the chain. We show that then $v_i^\flat\sigma \hookrightarrow^*_{\mathcal{R},\mu} C[v_j^\flat\sigma]$ for an active context C. Here, $(f^\sharp(t_1,\ldots,t_n))^\flat = f(t_1,\ldots,t_n)$ for all $f \in \mathcal{D}$. Doing this for all such DPs implies that there is an infinite reduction w.r.t. $(\mathcal{R},\mu)$.

If $v_i \to w_i \in \mathsf{DP}_o(R,\mu)$ then the claim is trivial, because then $j = i+1$ and $v_i^\flat\sigma \hookrightarrow_{\mathcal{R},\mu} C[w_i^\flat\sigma] \hookrightarrow^*_{\mathcal{R},\mu} C[v_{i+1}^\flat\sigma]$ for some active context C.

Otherwise, $v_i \to w_i$ has the form $v_i \to \mathsf{U}(x)$. Then $v_i^\flat\sigma \hookrightarrow_{\mathcal{R},\mu} C_1[\sigma(x)]$ for an active context C_1. Moreover, $\mathsf{U}(\sigma(x))$ reduces to $\mathsf{U}(\delta(t))$ for a hidden term t and

a δ by removing hiding contexts. Since hiding contexts are active, $\sigma(x) = C_2[\delta(t)]$ for an active context C_2. Finally, $t^\sharp\delta \xhookrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu} v_j\sigma$ and thus, $t\delta \xhookrightarrow{>\varepsilon}{}^*_{\mathcal{R},\mu} v_j^\flat\sigma$. By defining $C = C_1[C_2]$, we get $v_i^\flat\sigma \hookrightarrow^+_{\mathcal{R},\mu} C[v_j^\flat\sigma]$. □

4 CS Dependency Pair Framework

By Thm. 12, (innermost) termination of a CS-TRS is equivalent to absence of infinite (innermost) chains. For ordinary rewriting, the *DP framework* is the most recent and powerful collection of methods to prove absence of infinite chains automatically. Due to our new notion of (non-collapsing) CS-DPs, adapting the DP framework to the context-sensitive case now becomes much easier.[8]

In the DP framework, termination techniques operate on *DP problems* instead of TRSs. Def. 13 adapts this notion to context-sensitive rewriting.

Definition 13 (CS-DP Problem and Processor). *A* CS-DP problem *is a tuple $(\mathcal{P}, \mathcal{R}, \mu, e)$, where $\mathcal{P}$ and $\mathcal{R}$ are TRSs, μ is a replacement map, and $e \in \{\mathbf{t}, \mathbf{i}\}$ is a flag that stands for* **t**ermination *or* **i**nnermost termination. *We also call $(\mathcal{P}, \mathcal{R}, \mu)$-chains "$(\mathcal{P}, \mathcal{R}, \mu, \mathbf{t})$-chains" and we call innermost $(\mathcal{P}, \mathcal{R}, \mu)$-chains "$(\mathcal{P}, \mathcal{R}, \mu, \mathbf{i})$-chains". A* CS-DP problem $(\mathcal{P}, \mathcal{R}, \mu, e)$ *is* finite *if there is no infinite $(\mathcal{P}, \mathcal{R}, \mu, e)$-chain.*

A CS-DP processor *is a function Proc that takes a CS-DP problem as input and returns a possibly empty set of CS-DP problems. The processor Proc is* sound *if a CS-DP problem d is finite whenever all problems in $Proc(d)$ are finite.*

For a CS-TRS $(\mathcal{R}, \mu)$, the termination proof starts with the *initial DP problem* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu, e)$ where e depends on whether one wants to prove termination or innermost termination. Then sound DP processors are applied repeatedly. If the final processors return empty sets, then (innermost) termination is proved. Since innermost termination is usually easier to show than full termination, one should use $e = \mathbf{i}$ whenever possible. As shown in [12], termination and innermost termination coincide for CS-TRSs $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is *orthogonal* (i.e., left-linear and without critical pairs). So $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu, \mathbf{i})$ would be the initial DP problem for Ex. 1, even when proving full termination. In Sect. 4.1 - 4.3, we recapitulate 3 important DP processors and extend them to context-sensitive rewriting.

4.1 Dependency Graph Processor

The first processor decomposes a DP problem into several sub-problems. To this end, one determines which pairs can follow each other in chains by constructing a *dependency graph*. In contrast to related definitions for collapsing CS-DPs in [1,4], Def. 14 is analogous to the corresponding definition for non-CS rewriting.

Definition 14 (CS-Dependency Graph). *For a CS-DP problem $(\mathcal{P}, \mathcal{R}, \mu, e)$, the nodes of the* $(\mathcal{P}, \mathcal{R}, \mu, e)$-dependency graph *are the pairs of $\mathcal{P}$, and there is an arc from $v \to w$ to $s \to t$ iff $v \to w, s \to t$ is a $(\mathcal{P}, \mathcal{R}, \mu, e)$-chain.*

[8] For this reason, we omitted the proofs in this section and refer to [5] for all proofs.

Example 15. Fig. 1 shows the dependency graph for Ex. 1, for both $e \in \{\mathbf{t}, \mathbf{i}\}$.[9]

A set $\mathcal{P}' \neq \varnothing$ of DPs is a *cycle* if for every $v \to w, s \to t \in \mathcal{P}'$, there is a non-empty path from $v \to w$ to $s \to t$ traversing only pairs of $\mathcal{P}'$. A cycle $\mathcal{P}'$ is a *strongly connected component* ("SCC") if $\mathcal{P}'$ is not a proper subset of another cycle.

One can prove termination separately for each SCC. Thus, the following processor (whose soundness is obvious and completely analogous to the non-context-sensitive case) modularizes termination proofs.

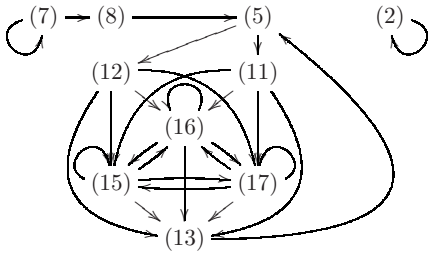


Fig. 1. Dependency graph for Ex. 1

Theorem 16 (CS-Dependency Graph Processor). *For* $d = (\mathcal{P}, \mathcal{R}, \mu, e)$, *let* $Proc(d) = \{(\mathcal{P}_1, \mathcal{R}, \mu, e), \ldots, (\mathcal{P}_n, \mathcal{R}, \mu, e)\}$, *where* $\mathcal{P}_1, \ldots, \mathcal{P}_n$ *are the SCCs of the* $(\mathcal{P}, \mathcal{R}, \mu, e)$*-dependency graph. Then Proc is sound.*

Example 17. The graph in Fig. 1 has the three SCCs $\mathcal{P}_1 = \{(2)\}$, $\mathcal{P}_2 = \{(7)\}$, $\mathcal{P}_3 = \{(5), (11)\text{-}(13), (15)\text{-}(17)\}$. *Thus, the initial DP problem* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu, \mathbf{i})$ *is transformed into the new problems* $(\mathcal{P}_1, \mathcal{R}, \mu, \mathbf{i})$, $(\mathcal{P}_2, \mathcal{R}, \mu, \mathbf{i})$, $(\mathcal{P}_3, \mathcal{R}, \mu, \mathbf{i})$.

As in the non-context-sensitive setting, the CS-dependency graph is not computable and thus, one has to use estimations to over-approximate the graph. For example, [1,4] adapted the estimation of [6] that was originally developed for ordinary rewriting: $\textsc{Cap}^\mu(t)$ replaces all active subterms of t with defined root symbol by different fresh variables. Multiple occurrences of the same such subterm are also replaced by pairwise different variables. $\textsc{Ren}^\mu(t)$ replaces all active occurrences of variables in t by different fresh variables (i.e., no variable occurs at several active positions in $\textsc{Ren}^\mu(t)$). So $\textsc{Ren}^\mu(\textsc{Cap}^\mu(\mathsf{IF}(\mathsf{gt}(y, \mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x))) = \textsc{Ren}^\mu(\mathsf{IF}(z, \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)) = \mathsf{IF}(z', \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)$.

To estimate the CS-dependency graph in the case $e = \mathbf{t}$, one draws an arc from $v \to w$ to $s \to t$ whenever $\textsc{Ren}^\mu(\textsc{Cap}^\mu(w))$ and s unify.[10] If $e = \mathbf{i}$, then one can modify $\textsc{Cap}^\mu$ and $\textsc{Ren}^\mu$ by taking into account that instantiated subterms at active positions of the left-hand side must be in normal form, cf. [4]. $\textsc{Cap}^\mu_v(w)$ is like $\textsc{Cap}^\mu(w)$, but the replacement of subterms of w by fresh variables is not done if the subterms also occur at active positions of v. Similarly, $\textsc{Ren}^\mu_v(w)$ is like $\textsc{Ren}^\mu(w)$, but the renaming of variables in w is not done if the variables

[9] To improve readability, we omitted nodes (6) and (14) from the graph. There are arcs from the nodes (8) and (13) to (6) and from all nodes (11), (12), (15), (16), (17) to (14). But (6) and (14) have no outgoing arcs and thus, they are not on any cycle.

[10] Here (and also later in the instantiation processor of Sect. 4.3), we always assume that $v \to w$ and $s \to t$ are renamed apart to be variable-disjoint.

also occur active in v. Now we draw an arc from $v \to w$ to $s \to t$ whenever $\text{REN}_v^\mu(\text{CAP}_v^\mu(w))$ and s unify by an mgu θ where $v\theta$ and $s\theta$ are in normal form.[11]

It turns out that for the TRS of Ex. 1, the resulting estimated dependency graph is identical to the "real" graph in Fig. 1.

4.2 Reduction Pair Processor

There are several processors to simplify DP problems by applying suitable *well-founded orders* (e.g., the *reduction pair processor* [17,21], the *subterm criterion processor* [22], etc.). Due to the absence of collapsing DPs, most of these processors are now straightforward to adapt to the context-sensitive setting. In the following, we present the reduction pair processor with *usable rules*, because it is the only processor whose adaption is more challenging. (The adaption is similar to the one in [4,20] for the CS-DPs of Def. 2.)

To prove that a DP problem is finite, the reduction pair processor generates constraints which should be satisfied by a *μ-reduction pair* $(\succsim, \succ)$ [1]. Here, $\succsim$ is a stable μ-monotonic quasi-order, $\succ$ is a stable well-founded order, and $\succsim$ and $\succ$ are compatible (i.e., $\succ \circ \succsim \subseteq \succ$ or $\succsim \circ \succ \subseteq \succ$). Here, *$\mu$-monotonicity* means that $s_i \succsim t_i$ implies $f(s_1, ..., s_i, ..., s_n) \succsim f(s_1, ..., t_i, ..., s_n)$ whenever $i \in \mu(f)$.

For a DP problem $(\mathcal{P}, \mathcal{R}, \mu, e)$, the generated constraints ensure that some rules in $\mathcal{P}$ are strictly decreasing (w.r.t. $\succ$) and all remaining rules in $\mathcal{P}$ and $\mathcal{R}$ are weakly decreasing (w.r.t. $\succsim$). Requiring $\ell \succsim r$ for all $\ell \to r \in \mathcal{R}$ ensures that in a chain $s_1 \to t_1, s_2 \to t_2, ...$ with $t_i\sigma \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}\sigma$, we have $t_i\sigma \succsim s_{i+1}\sigma$ for all i. Hence, if a reduction pair satisfies the constraints, then one can delete the strictly decreasing pairs from $\mathcal{P}$ as they cannot occur infinitely often in chains.

To improve this idea, it is desirable to require only a weak decrease of *certain* instead of *all* rules. In the non-context-sensitive setting, when proving innermost termination, it is sufficient if just the *usable rules* are weakly decreasing [6]. The same is true when proving full termination, provided that $\succsim$ is *$\mathcal{C}_\varepsilon$-compatible*, i.e., $\mathsf{c}(x, y) \succsim x$ and $\mathsf{c}(x, y) \succsim y$ holds for a fresh function symbol c [17,22].

For a term containing a symbol f, all f-rules are *usable*. Moreover, if the f-rules are usable and f *depends on* h (denoted $f \blacktriangleright_\mathcal{R} h$) then the h-rules are usable as well. Here, $f \blacktriangleright_\mathcal{R} h$ if $f = h$ or if there is a symbol g with $g \blacktriangleright_\mathcal{R} h$ and g occurs in the right-hand side of an f-rule. The usable rules of a DP problem are defined to be the usable rules of the right-hand sides of the DPs.

[11] These estimations can be improved further by adapting existing refinements to the context-sensitive case. However, different to the non-context-sensitive case, for $e = \mathsf{i}$ it is not sufficient to check only for unification of $\text{CAP}_v^\mu(w)$ and s (i.e., renaming variables with REN_v^μ is also needed). This can be seen from the non-innermost terminating CS-TRS $(\mathcal{R}, \mu)$ from [4, Ex. 8] with $\mathcal{R} = \{\mathsf{f}(\mathsf{s}(x), x) \to \mathsf{f}(x, x), \mathsf{a} \to \mathsf{s}(\mathsf{a})\}$ and $\mu(\mathsf{f}) = \{1\}$, $\mu(\mathsf{s}) = \varnothing$. Clearly, $\text{CAP}^\mu_{\mathsf{F}(\mathsf{s}(x),x)}(\mathsf{F}(x, x)) = \mathsf{F}(x, x)$ does not unify with $\mathsf{F}(\mathsf{s}(y), y)$. In contrast, $\text{REN}^\mu_{\mathsf{F}(\mathsf{s}(x),x)}(\text{CAP}^\mu_{\mathsf{F}(\mathsf{s}(x),x)}(\mathsf{F}(x, x))) = \mathsf{F}(x', x)$ unifies with $\mathsf{F}(\mathsf{s}(y), y)$. Thus, without using $\text{REN}^\mu_{\mathsf{F}(\mathsf{s}(x),x)}$ one would conclude that the dependency graph has no cycle and wrongly prove (innermost) termination.

As in [4,20], Def. 18 adapts[12] the concept of usable rules to the CS setting, resulting in $\mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu)$. But as shown in [20], for CS rewriting it is also helpful to consider an alternative definition of "dependence" $\rhd_{\mathcal{R},\mu}$ where f also depends on symbols from *left-hand sides* of f-rules. Let $\mathcal{F}^{\mu}(t)$ (resp. $\mathcal{F}^{\not\mu}(t)$) contain all function symbols occurring at active (resp. inactive) positions of a term t.

Definition 18 (CS-Usable Rules). *Let* $Rls(f) = \{\ell \to r \in \mathcal{R} \mid \mathrm{root}(\ell) = f\}$. *For any symbols* f, h *and CS-TRS* $(\mathcal{R},\mu)$, *let* $f \blacktriangleright_{\mathcal{R},\mu} h$ *if* $f = h$ *or if there is a symbol* g *with* $g \blacktriangleright_{\mathcal{R},\mu} h$ *and a rule* $\ell \to r \in Rls(f)$ *with* $g \in \mathcal{F}^{\mu}(r)$. *Let* $f \rhd_{\mathcal{R},\mu} h$ *if* $f = h$ *or if there is a symbol* g *with* $g \rhd_{\mathcal{R},\mu} h$ *and a rule* $\ell \to r \in Rls(f)$ *with* $g \in \mathcal{F}^{\not\mu}(\ell) \cup \mathcal{F}(r)$. *We define two forms of* usable rules*:*

$$\mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu) = \bigcup_{s\to t\in\mathcal{P}, f\in\mathcal{F}^{\mu}(t), f\blacktriangleright_{\mathcal{R},\mu} g} Rls(g)$$
$$\mathcal{U}^{\rhd}(\mathcal{P},\mathcal{R},\mu) = \bigcup_{s\to t\in\mathcal{P}, f\in\mathcal{F}^{\not\mu}(s)\cup\mathcal{F}(t), f\rhd_{\mathcal{R},\mu} g} Rls(g) \cup \bigcup_{\ell\to r\in\mathcal{R}, f\in\mathcal{F}^{\not\mu}(r), f\rhd_{\mathcal{R},\mu} g} Rls(g)$$

Example 19. We continue Ex. 17. $\mathcal{U}^{\blacktriangleright}(\mathcal{P}_1,\mathcal{R},\mu) = \varnothing$ *for* $\mathcal{P}_1 = \{(2)\}$, *since there is no defined symbol at an active position in the right-hand side* $\mathsf{GT}(x,y)$ *of (2). For* $\mathcal{P}_2 = \{(7)\}$, $\mathcal{U}^{\blacktriangleright}(\mathcal{P}_2,\mathcal{R},\mu)$ *are the* minus*-,* if*-, and* gt*-rules, since* minus *occurs at an active position in* $\mathsf{D}(\mathsf{minus}(x,y),\mathsf{s}(y))$ *and* minus *depends on* if *and* gt. *For* $\mathcal{P}_3 = \{(5),(11)\text{-}(13),(15)\text{-}(17)\}$, $\mathcal{U}^{\blacktriangleright}(\mathcal{P}_3,\mathcal{R},\mu)$ *are the* gt*- and* p*-rules, as* gt *and* p *are the only defined symbols at active positions of right-hand sides in* $\mathcal{P}_3$.

In contrast, all $\mathcal{U}^{\rhd}(\mathcal{P}_i,\mathcal{R},\mu)$ *contain all rules except the* div*-rules, as* minus *and* p *are root symbols of hidden terms and* minus *depends on* if *and* gt.

As shown in [4,20], the direct adaption of the usable rules to the context-sensitive case (i.e., $\mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu)$) can only be used for *conservative* CS-TRSs (if $e = \mathbf{i}$) resp. for *strongly conservative* CS-TRSs (if $e = \mathbf{t}$).[13] Let $\mathcal{V}^{\mu}(t)$ (resp. $\mathcal{V}^{\not\mu}(t)$) be all variables occurring at active (resp. inactive) positions of a term t.

Definition 20 (Conservative and Strongly Conservative). *A CS-TRS* $(\mathcal{R},\mu)$ *is* conservative *iff* $\mathcal{V}^{\mu}(r) \subseteq \mathcal{V}^{\mu}(\ell)$ *for all rules* $\ell \to r \in \mathcal{R}$. *It is* strongly conservative *iff it is conservative and moreover,* $\mathcal{V}^{\mu}(\ell) \cap \mathcal{V}^{\not\mu}(\ell) = \varnothing$ *and* $\mathcal{V}^{\mu}(r) \cap \mathcal{V}^{\not\mu}(r) = \varnothing$ *for all rules* $\ell \to r \in \mathcal{R}$.

Now we can define the reduction pair processor.

Theorem 21 (CS-Reduction Pair Processor). *Let* $(\succsim,\succ)$ *be a* μ*-reduction pair. For a CS-DP Problem* $d = (\mathcal{P},\mathcal{R},\mu,e)$, *the result of Proc*$(d)$ *is*

- $\{(\mathcal{P}\setminus\succ,\mathcal{R},\mu,e)\}$, *if* $\mathcal{P} \subseteq (\succ \cup \succsim)$ *and at least one of the following holds:*

[12] The adaptions can also be extended to refined definitions of usable rules [15,17].

[13] The corresponding counterexamples in [4,20] show that these restrictions are still necessary for our new notion of CS-DPs. In cases where one cannot use $\mathcal{U}^{\blacktriangleright}$, one can also attempt a termination proof where one drops the replacement map, i.e., where one regards the ordinary TRS $\mathcal{R}$ instead of the CS-TRS $(\mathcal{R},\mu)$. This may be helpful, since $\mathcal{U}^{\rhd}$ is not necessarily a subset of the non-context-sensitive usable rules, as a function symbol f also $\rhd$-depends on symbols from *left*-hand sides of f-rules.

(i) $\mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu) \subseteq \succsim$, $\mathcal{P} \cup \mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu)$ *is strongly conservative,* $\succsim$ *is* $\mathcal{C}_\varepsilon$*-compatible*
(ii) $\mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu) \subseteq \succsim$, $\mathcal{P} \cup \mathcal{U}^{\blacktriangleright}(\mathcal{P},\mathcal{R},\mu)$ *is conservative,* $e = \mathbf{i}$
(iii) $\mathcal{U}^{\triangleright}(\mathcal{P},\mathcal{R},\mu) \subseteq \succsim$, $\succsim$ *is* $\mathcal{C}_\varepsilon$*-compatible*
(iv) $\mathcal{R} \subseteq \succsim$

- $\{d\}$, *otherwise.*

Then Proc is sound.

Example 22. As $\mathcal{U}^{\blacktriangleright}(\mathcal{P}_1,\mathcal{R},\mu) = \varnothing$ *and* $\mathcal{P}_1 = \{(2)\}$ *is even strongly conservative, by Thm. 21 (i) or (ii) we only have to orient (2), which already works with the embedding order. So* $(\mathcal{P}_1,\mathcal{R},\mu,\mathbf{i})$ *is transformed to the empty set of DP problems.*

For $\mathcal{P}_2 = \{(7)\}$, $\mathcal{U}^{\blacktriangleright}(\mathcal{P}_2,\mathcal{R},\mu)$ *contains the* if*-rules which are not conservative. Hence, we use Thm. 21 (iii) with a reduction pair based on the following max-polynomial interpretation [10]:* $[\mathsf{D}(x,y)] = [\mathsf{minus}(x,y)] = [\mathsf{p}(x)] = x$, $[\mathsf{s}(x)] = x+1$, $[\mathsf{if}(x,y,z)] = \max(y,z)$, $[\mathsf{0}] = [\mathsf{gt}(x,y)] = [\mathsf{true}] = [\mathsf{false}] = 0$. *Then the DP (7) is strictly decreasing and all rules from* $\mathcal{U}^{\triangleright}(\mathcal{P}_2,\mathcal{R},\mu)$ *are weakly decreasing. Thus, the processor also transforms* $(\mathcal{P}_2,\mathcal{R},\mu,\mathbf{i})$ *to the empty set of DP problems.*

Finally, we regard $\mathcal{P}_3 = \{(5),(11)\text{-}(13),(15)\text{-}(17)\}$ *where we use Thm. 21 (iii) with the interpretation* $[\mathsf{M}(x,y)] = [\mathsf{minus}(x,y)] = x+y+1$, $[\mathsf{IF}(x,y,z)] = [\mathsf{if}(x,y,z)] = \max(y,z)$, $[\mathsf{U}(x)] = [\mathsf{p}(x)] = [\mathsf{s}(x)] = x$, $[\mathsf{0}] = [\mathsf{gt}(x,y)] = [\mathsf{true}] = [\mathsf{false}] = 0$. *Then the DPs (16) and (17) are strictly decreasing, whereas all other DPs from* $\mathcal{P}_3$ *and all rules from* $\mathcal{U}^{\triangleright}(\mathcal{P}_3,\mathcal{R},\mu)$ *are weakly decreasing. So the processor results in the DP problem* $(\{(5),(11)\text{-}(13),(15)\},\mathcal{R},\mu,\mathbf{i})$.

Next we apply $[\mathsf{M}(x,y)] = [\mathsf{minus}(x,y)] = x+1$, $[\mathsf{IF}(x,y,z)] = \max(y,z+1)$, $[\mathsf{if}(x,y,z)] = \max(y,z)$, $[\mathsf{U}(x)] = [\mathsf{p}(x)] = [\mathsf{s}(x)] = x$, $[\mathsf{0}] = [\mathsf{gt}(x,y)] = [\mathsf{true}] = [\mathsf{false}] = 0$. *Now (12) is strictly decreasing and all other remaining DPs and usable rules are weakly decreasing. Removing (12) yields* $(\{(5),(11),(13),(15)\},\mathcal{R},\mu,\mathbf{i})$.

Thm. 21 (iii) and (iv) are a significant improvement over previous reduction pair processors [1,2,4,20] for the CS-DPs from Def. 2. The reason is that all previous CS-reduction pair processors require that the context-sensitive subterm relation is contained in $\succsim$ (i.e., $\trianglerighteq_\mu \subseteq \succsim$) whenever there are collapsing DPs. This is a very hard requirement which destroys one of the main advantages of the DP method (i.e., the possibility to filter away arbitrary arguments).[14] With our new non-collapsing CS-DPs, this requirement is no longer needed.

Example 23. If one requires $\trianglerighteq_\mu \subseteq \succsim$*, then the reduction pair processor would fail for Ex. 1, since then one cannot make the DP (7) strictly decreasing. The reason is that due to* $2 \in \mu(\mathsf{minus})$, $\trianglerighteq_\mu \subseteq \succsim$ *implies* $\mathsf{minus}(x,y) \succsim y$. *So one cannot "filter away" the second argument of* minus. *But then a strict decrease of DP (7) together with* μ*-monotonicity of* $\succsim$ *implies* $\mathsf{D}(\mathsf{s}(x),\mathsf{s}(\mathsf{s}(x))) \succ \mathsf{D}(\mathsf{minus}(x,\mathsf{s}(x)),\mathsf{s}(\mathsf{s}(x))) \succsim \mathsf{D}(\mathsf{s}(x),\mathsf{s}(\mathsf{s}(x)))$, *in contradiction to the well-foundedness of* $\succ$.

[14] Moreover, previous CS-reduction pair processors also require $f(x_1,\ldots,x_n) \succsim f^\sharp(x_1,\ldots,x_n)$ for all $f \in \mathcal{D}$ or $f(x_1,\ldots,x_n) \succ f^\sharp(x_1,\ldots,x_n)$ for all $f \in \mathcal{D}$. This requirement also destroys an important feature of the DP method, i.e., that tuple symbols $f^\sharp$ can be treated independently from the original corresponding symbols f. This feature often simplifies the search for suitable reduction pairs considerably.

4.3 Transforming Context-Sensitive Dependency Pairs

To increase the power of the DP method, there exist several processors to transform a DP into new pairs (e.g., *narrowing*, *rewriting*, *instantiating*, or *forward instantiating* DPs [17]). We now adapt the *instantiation* processor to the context-sensitive setting. Similar adaptions can also be done for the other processors.[15]

The idea of this processor is the following. For a DP $s \to t$, we investigate which DPs $v \to w$ can occur before $s \to t$ in chains. To this end, we use the same estimation as for dependency graphs in Sect. 4.1, i.e., we check whether there is an mgu θ of $\text{REN}^\mu(\text{CAP}^\mu(w))$ and s if $e = \mathbf{t}$ and analogously for $e = \mathbf{i}$.[16] Then we replace $s \to t$ by the new DPs $s\theta \to t\theta$ for all such mgu's θ. This is sound since in any chain $\ldots, v \to w, s \to t, \ldots$ where an instantiation of w reduces to an instantiation of s, one could use the new DP $s\theta \to t\theta$ instead.

Theorem 24 (CS-Instantiation Processor). *Let $\mathcal{P}' = \mathcal{P} \uplus \{s \to t\}$. For $d = (\mathcal{P}', \mathcal{R}, \mu, e)$, let the result of Proc$(d)$ be $(\mathcal{P} \cup \overline{\mathcal{P}}, \mathcal{R}, \mu, e)$ where*

- $\overline{\mathcal{P}} = \{s\theta \to t\theta \mid \theta = mgu(\text{REN}^\mu(\text{CAP}^\mu(w)), s), v \to w \in \mathcal{P}'\}$, *if* $e = \boldsymbol{t}$
- $\overline{\mathcal{P}} = \{s\theta \to t\theta \mid \theta = mgu(\text{REN}^\mu_v(\text{CAP}^\mu_v(w)), s), v \to w \in \mathcal{P}', s\theta, v\theta \text{ normal}\}$, *if* $e = \boldsymbol{i}$

Then Proc is sound.

Example 25. For the TRS of Ex. 1, we still had to solve the problem $(\{(5), (11), (13), (15)\}, \mathcal{R}, \mu, \mathbf{i})$, *cf. Ex. 22. DP (11) has the variable-renamed left-hand side* $\mathsf{IF}(\mathsf{true}, x', y')$. *So the only DP that can occur before (11) in chains is* (5) *with the right-hand side* $\mathsf{IF}(\mathsf{gt}(y, \mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)$. *Recall* $\text{REN}^\mu(\text{CAP}^\mu(\mathsf{IF}(\mathsf{gt}(y, \mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x))) = \mathsf{IF}(z', \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)$, *cf. Sect. 4.1. So the mgu is* $\theta = [z'/\mathsf{true},\ x'/\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)),\ y'/x]$. *Hence, we can replace (11) by*

$$\mathsf{IF}(\mathsf{true}, \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x) \to \mathsf{U}(\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))) \tag{20}$$

Here the CS variant of the instantiation processor is advantageous over the non-CS one which uses CAP *instead of* CAP^μ, *where* CAP *replaces all subterms with defined root (e.g.,* $\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))$*) by fresh variables. So the non-CS processor would not help here as it only generates a variable-renamed copy of (11).*

When re-computing the dependency graph, there is no arc from (20) to (15) as $\mu(\mathsf{U}) = \varnothing$. *So the DP problem is decomposed into* $(\{(15)\}, \mathcal{R}, \mu, \mathbf{i})$ *(which is easily solved by the reduction pair processor) and* $(\{(5), (20), (13)\}, \mathcal{R}, \mu, \mathbf{i})$.

Now we apply the reduction pair processor again with the following rational *polynomial interpretation [11]:* $[\mathsf{M}(x, y)] = \frac{3}{2}x + \frac{1}{2}y$, $[\mathsf{minus}(x, y)] = 2x + \frac{1}{2}y$, $[\mathsf{IF}(x, y, z)] = \frac{1}{2}x + y + \frac{1}{2}z$, $[\mathsf{if}(x, y, z)] = \frac{1}{2}x + y + z$, $[\mathsf{U}(x)] = x$, $[\mathsf{p}(x)] = [\mathsf{gt}(x, y)] = \frac{1}{2}x$, $[\mathsf{s}(x)] = 2x + 2$, $[\mathsf{true}] = 1$, $[\mathsf{false}] = [\mathsf{0}] = 0$. *Then (20) is strictly decreasing and can be removed, whereas all other remaining DPs and usable rules*

[15] In the papers on CS-DPs up to now, the only existing adaption of such a processor was the straightforward adaption of the *narrowing* processor in the case $e = \mathbf{t}$, cf. [2]. However, this processor would not help for the TRS of Ex. 1.

[16] The counterexample of [4, Ex. 8] in Footnote 11 again illustrates why REN^μ_v is also needed in the innermost case (whereas this is unnecessary for non-CS rewriting).

are weakly decreasing. A last application of the dependency graph processor then detects that there is no cycle anymore and thus, it returns the empty set of DP problems. Hence, termination of the TRS from Ex. 1 is proved. As shown in our experiments in Sect. 5, this proof can easily be performed automatically.

5 Experiments and Conclusion

We have developed a new notion of context-sensitive dependency pairs which improves significantly over previous notions. There are two main advantages:

(1) **Easier adaption of termination techniques to CS rewriting**
Now CS-DPs are very similar to DPs for ordinary rewriting and consequently, the existing powerful termination techniques from the DP framework can easily be adapted to context-sensitive rewriting. We have demonstrated this with some of the most popular DP processors in Sect. 4. Our adaptions subsume the existing earlier adaptions of the dependency graph [2], of the usable rules [20], and of the modifications for innermost rewriting [4], which were previously developed for the notion of CS-DPs from [1].

(2) **More powerful termination analysis for CS rewriting**
Due to the absence of collapsing CS-DPs, one does not have to impose extra restrictions anymore when extending the DP processors to CS rewriting, cf. Ex. 23. Hence, the power of termination proving is increased substantially.

To substantiate Claim (2), we performed extensive experiments. We implemented our new non-collapsing CS-DPs and all DP processors from this paper in the termination prover AProVE [16].[17] In contrast, the prover MU-TERM [3] uses the collapsing CS-DPs. Moreover, the processors for these CS-DPs are not formulated within the DP framework and thus, they cannot be applied in the same flexible and modular way. While MU-TERM was the most powerful tool for termination analysis of context-sensitive rewriting up to now (as demonstrated by the *International Competition of Termination Tools* 2007 [27]), due to our new notion of CS-DPs, now AProVE is substantially more powerful. For instance, AProVE easily proves termination of our leading example from Ex. 1, whereas MU-TERM fails. Moreover, we tested the tools on all 90 context-sensitive TRSs from the *Termination Problem Data Base* that was used in the competition. We used a time limit of 120 seconds for each example. Then MU-TERM can prove termination of 68 examples, whereas the new version of AProVE proves termination of 78 examples (including all 68 TRSs where MU-TERM is successful).[18] Since 4 examples are known to be non-terminating, at most 8 more of the 90 examples could potentially be detected as terminating. So due to the results of this paper, termination proving of context-sensitive rewriting has now become

[17] We also used the subterm criterion and forward instantiation processors, cf. Sect. 4.

[18] If AProVE is restricted to use exactly the same processors as MU-TERM, then it still succeeds on 74 examples. So its superiority is indeed mainly due to the new CS-DPs which enable an easy adaption of the DP framework to the CS setting.

very powerful. To experiment with our implementation and for details, we refer to `http://aprove.informatik.rwth-aachen.de/eval/CS-DPs/`.

References

1. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-sensitive dependency pairs. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 297–308. Springer, Heidelberg (2006)
2. Alarcón, B., Gutiérrez, R., Lucas, S.: Improving the context-sensitive dependency graph. In: Proc. PROLE 2006. ENTCS, vol. 188, pp. 91–103 (2007)
3. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving termination of context-sensitive rewriting with Mu-term. Pr. PROLE 2006. ENTCS, vol. 188, p. 105–115 (2007)
4. Alarcón, B., Lucas, S.: Termination of innermost context-sensitive rewriting using dependency pairs. In: Konev, B., Wolter, F. (eds.) FroCos 2007. LNCS, vol. 4720, pp. 73–87. Springer, Heidelberg (2007)
5. Alarcón, B., Emmes, F., Fuhs, C., Giesl, J., Gutiérrez, R., Lucas, S., Schneider-Kamp, P., Thiemann, R.: Improving context-sensitive dependency pairs. Technical Report AIB-2008-13 (2008), `http://aib.informatik.rwth-aachen.de/`
6. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236, 133–178 (2000)
7. Baader, F., Nipkow, T.: Term Rewriting and All That, Cambridge (1998)
8. Borralleras, C., Lucas, S., Rubio, A.: Recursive path orderings can be context-sensitive. In: Voronkov, A. (ed.) CADE 2002. LNCS, vol. 2392, pp. 314–331. Springer, Heidelberg (2002)
9. Dershowitz, N.: Termination by abstraction. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 1–18. Springer, Heidelberg (2004)
10. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: Maximal termination. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 110–125. Springer, Heidelberg (2008)
11. Fuhs, C., Navarro-Marset, R., Otto, C., Giesl, J., Lucas, S., Schneider-Kamp, P.: Search techniques for rational polynomial orders. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) AISC 2008, Calculemus 2008, and MKM 2008. LNCS (LNAI), vol. 5144, pp. 109–124. Springer, Heidelberg (2008)
12. Giesl, J., Middeldorp, A.: Innermost termination of context-sensitive rewriting. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 231–244. Springer, Heidelberg (2003)
13. Giesl, J., Middeldorp, A.: Transformation techniques for context-sensitive rewrite systems. Journal of Functional Programming 14(4), 379–427 (2004)
14. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS, vol. 3452, pp. 301–331. Springer, Heidelberg (2005)
15. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Gramlich, B. (ed.) FroCos 2005. LNCS (LNAI), vol. 3717, pp. 216–231. Springer, Heidelberg (2005)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the dependency pair framework. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS, vol. 4130, pp. 281–286. Springer, Heidelberg (2006)

17. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. Journal of Automatic Reasoning 37(3), 155–203 (2006)
18. Gramlich, B.: Generalized sufficient conditions for modular termination of rewriting. Appl. Algebra in Engineering, Comm. and Computing 5, 131–151 (1994)
19. Gramlich, B., Lucas, S.: Simple termination of context-sensitive rewriting. In: Proc. RULE 2002, pp. 29–41. ACM Press, New York (2002)
20. Gutiérrez, R., Lucas, S., Urbain, X.: Usable rules for context-sensitive rewrite systems. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 126–141. Springer, Heidelberg (2008)
21. Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. Information and Computation 199(1,2), 172–199 (2005)
22. Hirokawa, N., Middeldorp, A.: Tyrolean Termination Tool: techniques and features. Information and Computation 205(4), 474–511 (2007)
23. Lucas, S.: Context-sensitive computations in functional and functional logic programs. Journal of Functional and Logic Programming 1998(1), 1–61 (1998)
24. Lucas, S.: Context-sensitive rewriting strategies. Inf. Comp. 178(1), 293–343 (2002)
25. Lucas, S.: Polynomials for proving termination of context-sensitive rewriting. In: Walukiewicz, I. (ed.) FOSSACS 2004. LNCS, vol. 2987, pp. 318–332. Springer, Heidelberg (2004)
26. Lucas, S.: Proving termination of context-sensitive rewriting by transformation. Information and Computation 204(12), 1782–1846 (2006)
27. Marché, C., Zantema, H.: The termination competition. In: Baader, F. (ed.) RTA 2007. LNCS, vol. 4533, pp. 303–313. Springer, Heidelberg (2007)
28. Urbain, X.: Modular & incremental automated termination proofs. Journal of Automated Reasoning 32(4), 315–355 (2004)

18.6 Using *CSR* for Proving Innermost Termination of Rewriting

6. B. Alarcón, and S. Lucas. **Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting**. *Electronic Notes in Theoretical Computer Science*, 248:3-17, 2009. Selected papers from the 8th Spanish Conference on Programming and Languages, PROLE'08.

Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 248 (2009) 3–17
www.elsevier.com/locate/entcs

Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting

Beatriz Alarcón and Salvador Lucas [1,2]

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain

Abstract

Computational systems based on reducing expressions usually have a predefined reduction strategy to break down the nondeterminism which is inherent to reduction relations. The innermost strategy corresponds to call by value or *eager* computation, that is, the computational mechanism of several programming languages like Maude, OBJ, etc. where the arguments of a function call are always *evaluated* before calling the function. This strategy usually fails to terminate when nonterminating computations are possible in the programs and many eager programming languages also admit the *explicit* specification of a particular class of *strategy annotations* to (try to) avoid them. Context-Sensitive Rewriting provides an abstract model to describe and analyze the operational behavior of such programs. This paper aims at contributing to the development of appropriate techniques and tools for the verification of *program termination* in the aforementioned programming languages, so we focus on termination of innermost (context-sensitive) rewriting. We adapt the notion of usable argument introduced by Fernández to prove innermost termination by proving termination of context-sensitive rewriting. Thanks to our recent developments for proving termination of (innermost) context-sensitive rewriting using dependency pairs, now we can also relax monotonicity requirements for proving innermost termination of (context-sensitive) rewriting. We have implemented these new improvements in the termination tool MU-TERM and evaluated the results with some benchmarks.

Keywords: Dependency pairs, innermost rewriting, context-sensitive rewriting program analysis, termination.

1 Introduction

Most computational systems whose operational principle is based on reducing expressions (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) incorporate a predefined reduction strategy which is used to break down the nondeterminism which is inherent to reduction relations. Thus, every program will be executed according to that strategy. One of the most commonly used is the *innermost* strategy, in which only

[1] This work has been partially supported by the EU (FEDER) and the Spanish MICINN, under grant TIN2007-68093-C02-02 and HA 2006-2007, the Generalitat Valenciana under grant GVPRE/2008/113. Also it was partially supported by the Spanish MICINN under FPU grant AP2005-3399.

[2] Email: {balarcon,slucas}@dsic.upv.es

1571-0661/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.entcs.2009.07.055

innermost redexes are reduced. Here, by an innermost redex we mean a redex containing no other redex. The innermost strategy corresponds to call by value or *eager* computation, that is, the computational mechanism of several programming languages where the arguments of a function are always evaluated before the application of the function which use them. It is well-known, however, that programs written in eager programming languages frequently run into a nonterminating behavior if the programs have not carefully been written to avoid such problems. For this reason, the designers of such eager programming languages have also developed some features and language constructs aimed at giving the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to the arguments of the function symbols) have been used in eager programming languages such as Maude [7], OBJ2 [11], OBJ3 [16], and CafeOBJ [12] to introduce *replacement restrictions* which are able to (hopefully) avoid nontermination. Such languages admit the *explicit* specification of a particular class of *strategy annotations*, which (basically) are lists of integers associated to function symbols which specify the ordering in which the arguments are (eventually) evaluated in function calls. This very simple strategy language provides quite a powerful way to control the program execution. Due to its simplicity, such strategy annotations also provide a simple interface for understanding and eventually modifying the execution of programs. Context-sensitive rewriting (*CSR* [18,21]) provides an abstract model to describe and analyze the operational behavior of such programs, thus providing an appropriate basis for the development of program verification tools [8,19,20]. In *CSR*, a *replacement map* (i.e., a mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f of a signature $\mathcal{F}$) associates a subset of its argument indices to each function symbol. We use a replacement map to indicate the argument positions on which rewriting steps are allowed. In this way, we can *achieve* a terminating behavior by pruning (all) infinite rewrite sequences. A Term Rewriting System (TRS) $\mathcal{R}$ together with a replacement map μ is often called a CS-TRS (written $(\mathcal{R}, \mu)$). The research in this paper aims at contributing to the development of appropriate techniques and tools for the verification of *program termination* in the aforementioned programming languages. Our focus is on termination of *innermost context-sensitive rewriting* (i.e., the variant of *CSR* where only the deepest μ-replacing redexes are contracted). Techniques for proving termination of innermost *CSR* were first investigated in [13,19]. These papers, though, only consider *transformational* techniques, where the original CS-TRS $(\mathcal{R}, \mu)$ is transformed into a TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation which has been used) whose *innermost* termination implies the innermost termination of *CSR* for $(\mathcal{R}, \mu)$. In [5], we have extended the context-sensitive dependency pairs approach in [2,3] for proving termination of innermost *CSR*. Roughly speaking the (context-sensitive) dependency pairs associated to a (CS-)TRS are of a set of rewrite rules which are used together with the original ones to obtain an often easier proof of termination due to the possibility of applying a number of new auxiliary techniques, see, e.g., [15,17] for recent state-of-the-art accounts. As shown in [3], proofs of termination using context-sensitive dependency pairs (CS-DPs) are much more powerful and faster than any other technique for proving termination of *CSR*.

Dealing with innermost *CSR*, we have a similar situation, see [5].

The main topic we are going to develop in this work is the analysis and extension of Fernández's work [9]. In her paper, she noticed that, when dealing with proofs of innermost termination, requiring monotonicity of the orderings w.r.t. all arguments of function symbols is not always necessary. According to this, she showed that innermost termination of rewriting can be rephrased as a context-sensitive rewriting termination problem. She introduced the notion of *usable arguments*, which can be thought of as the argument positions on which innermost reductions take place. Then, Fernández showed that innermost termination of a TRS $\mathcal{R}$ can be proved by proving termination of the CS-TRS $(\mathcal{R}, \mu)$ which is obtained when $\mu(f)$ collects the *usable arguments* of f for each symbol f in the signature. We have implemented her techniques for the first time, and then we have investigated the practical use of her results. We have adapted Fernández ideas to deal with proofs of *innermost termination of CSR*; this was left as an open problem in [9].

After some preliminaries in Section 2, in Section 3 we summarize the last developments concerning context-sensitive rewriting and innermost context-sensitive rewriting using dependency pairs. In Section 4 we show how to adapt Fernández's criterion to relax monotonicity requirements when proving innermost termination of *CSR*. In Section 5 we apply this criterion to be used with dependency pairs in proofs of termination of innermost (context-sensitive) rewriting. Section 6 provides an experimental evaluation of our techniques for proving innermost termination of (context-sensitive) rewriting automatically. Finally, we conclude and comment on some future work.

2 Preliminaries

Relations.

A (strict) partial ordering $>$ is an irreflexive and transitive relation. We say that $>$ is well-founded if there is no infinite decreasing sequence with $>$. A quasi-ordering $\gtrsim$ is a transitive and reflexive relation.

Terms.

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote their concatenation as $p.q$. Positions are ordered by the standard prefix ordering $\leq$. If p is a position, and Q is a set of positions, then $p.Q = \{p.q \mid q \in Q\}$. We denote the topmost position by $\wedge$. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of nonvariable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$ while $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. The symbol labelling the root of t is denoted as $root(t)$.

Term rewriting.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A rule is collapsing if $r \in \mathcal{X}$. An instance σ of a left-hand side of a rewrite rule $l \to r$, written $\sigma(l)$, is a redex (**red**ucible **ex**presion). A TRS is a pair $\mathcal{R} = (\mathcal{F}, R)$ where R is a set of rewrite rules. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined functions*, where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Innermost rewriting.

A term is a normal form if it contains no redex. A substitution σ is normalized if $\sigma(x)$ is a normal form for all $x \in \mathcal{D}om(\sigma)$. A term $f(t_1, \ldots, t_k)$ is argument normalized if t_i is a normal form for all $1 \leq i \leq n$. An innermost redex is an argument normalized redex. A term s rewrites innermost to t, written $s \to_i t$, if $s \to t$ at position p and $s|_p$ is an innermost redex. Let $\mathcal{R}$ be a TRS. For any symbol f let $Rules(\mathcal{R}, f)$ be the set of rules $l \to r$ defining f and such that the left-hand sides l are argument normalized. For any term t the set of usable rules $\mathbf{U}(\mathcal{R}, t)$ is as follows:

$$\begin{aligned} \mathbf{U}(\mathcal{R}, x) &= \varnothing \\ \mathbf{U}(\mathcal{R}, f(t_1, \ldots, t_n)) &= Rules(\mathcal{R}, f) \cup \bigcup_{1 \leq i \leq ar(f)} \mathbf{U}(\mathcal{R}', t_i) \cup \bigcup_{l \to r \in Rules(\mathcal{R}, f)} \mathbf{U}(\mathcal{R}', r) \end{aligned}$$

where $\mathcal{R}' = \mathcal{R} - Rules(\mathcal{R}, f)$.

Context-sensitive rewriting.

A mapping $\mu : \mathcal{F} \to \mathcal{P}(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1, \ldots, ar(f)\}$ [18]. Let $M_{\mathcal{F}}$ be the set of all $\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F}, R)$). A binary relation R on terms is μ-monotonic if $t\ \mathsf{R}\ s$ implies $f(t_1, \ldots, t_{i-1}, t, \ldots, t_k)\ \mathsf{R}\ f(t_1, \ldots, t_{i-1}, s, \ldots, t_k)$ for all $f \in \mathcal{F}$, $i \in \mu(f)$ and $t, s, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. The set of *$\mu$-replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i)$, if $t \notin \mathcal{X}$. The set of *μ-replacing* variables of t is $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$. A rule $l \to r$ is μ-conservative if $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$. The μ-replacing subterm relation $\trianglerighteq_{\mu}$ is given by $t \trianglerighteq_{\mu} s$ if there is $p \in \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. We write $t \triangleright_{\mu} s$ if $t \trianglerighteq_{\mu} s$ and $t \neq s$ and say that s is a strict μ-replacing subterm of t. We write $t \triangleright_{\not\mu} s$ to denote that s is a non-μ-replacing (hence strict) subterm of t: $t \triangleright_{\not\mu} s$ if there is $p \in \mathcal{P}os(t) - \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. In *context-sensitive rewriting* (*CSR* [18]), we (only) contract *μ-replacing* redexes: s μ-rewrites to t, written $s \hookrightarrow_{\mu} t$ (or $s \hookrightarrow_{\mathcal{R},\mu} t$ and even $s \hookrightarrow t$, if $\mathcal{R}$ and μ are clear from the context), if $s \xrightarrow{p}_{\mathcal{R}} t$ and $p \in \mathcal{P}os^{\mu}(s)$. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mu}$ is terminating. Termination of *CSR* is fully captured by the so-called μ-reduction orderings, i.e., well-founded, stable orderings $\sqsupset$ which are μ-monotonic. A term t is μ-terminating if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mu} t_2 \hookrightarrow_{\mu} \cdots \hookrightarrow_{\mu} t_n \hookrightarrow_{\mu} \cdots$ starting from t. A term t *μ-narrows* to a term s (written $t \rightsquigarrow_{\mathcal{R},\mu,\theta} s$), if there is a nonvariable μ-replacing position $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$ and a rule $l \to r$ in $\mathcal{R}$ (sharing no variable with t) such that $t|_p$ and l unify with most general unifier θ and $s = \theta(t[r]_p)$. Then, we say that

t is μ-narrowable. A μ-normal form is a term which cannot be μ-rewritten. Let $\mathsf{NF}_\mu(\mathcal{R})$ (or just NF_μ if no confusion arises) be the set of μ-normal forms of a TRS $\mathcal{R}$. A substitution σ is *μ-normalized* if $\sigma(x)$ is a μ-normal form for all $x \in \mathcal{D}om(\sigma)$. A term $t = f(t_1, \ldots, t_k)$ is *argument μ-normalized* if t_i is a μ-normal form for all $i \in \mu(f)$. A *μ-innermost redex* is an argument μ-normalized redex.

A term s innermost μ-rewrites to t, written $s \hookrightarrow_{\mathsf{i}} t$, if $s \xrightarrow{p}_{\mathcal{R}} t$, $p \in \mathcal{P}os^\mu(s)$, and $s|_p$ is a μ-innermost redex. A TRS $\mathcal{R}$ is innermost μ-terminating if $\hookrightarrow_{\mu,\mathsf{i}}$ is terminating. We write $s \hookrightarrow^!_{\mathcal{R},\mu,\mathsf{i}} t$ if $s \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} t$ and $t \in \mathsf{NF}_\mu$. A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_\mathcal{R}$ is often called a CS-TRS.

Dependency pairs.

Given a TRS $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ a new TRS $\mathsf{DP}(\mathcal{R}) = (\mathcal{F}^\sharp, D(R))$ of *dependency pairs* for $\mathcal{R}$ is given as follows: if $f(t_1, \ldots, t_m) \to r \in R$ and $r = C[g(s_1, \ldots, s_n)]$ for some defined symbol $g \in \mathcal{D}$ and $s_1, \ldots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, then $f^\sharp(t_1, \ldots, t_m) \to g^\sharp(s_1, \ldots, s_n) \in D(R)$, where $f^\sharp$ and $g^\sharp$ are new fresh symbols (called *tuple* symbols) associated to defined symbols f and g respectively [6]. Let $\mathcal{D}^\sharp$ be the set of tuple symbols associated to symbols in $\mathcal{D}$ and $\mathcal{F}^\sharp = \mathcal{F} \cup \mathcal{D}^\sharp$. As usual, for $t = f(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we write $t^\sharp$ to denote the *marked* term $f^\sharp(t_1, \ldots, t_k)$. Given $T \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$, $T^\sharp$ denotes $\{t^\sharp \mid t \in T\}$. For the sake of readability, capital letters denote marked symbols in examples.

Reduction pairs.

Given a signature $\mathcal{F}$, a reduction pair $(\succeq, \sqsupset)$ for terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ consists of a stable and monotonic quasi-ordering $\succeq$ on terms, and a stable and well-founded ordering $\sqsupset$ satisfying either $\succeq \circ \sqsupset \;\subseteq\; \sqsupset$ or $\sqsupset \circ \succeq \;\subseteq\; \sqsupset$. Note that *monotonicity is not required* for $\sqsupset$. A μ-reduction pair is a reduction pair $(\succeq, \sqsupset)$ where the quasi-ordering $\succeq$ is μ-monotonic (instead of monotonic).

3 Context-sensitive dependency pairs

In the following, we write $\textsc{Narr}^\mu(t)$ to indicate that t is μ-narrowable (w.r.t. the intended TRS $\mathcal{R}$). We consider a function $\textsc{Ren}^\mu$ which *independently* renames all *occurrences* of μ-replacing variables within a term t by using new fresh variables which are not in $\mathcal{V}ar(t)$:

- $\textsc{Ren}^\mu(x) = y$ if x is a variable, where y is intended to be a fresh new variable which has not yet been used (we could think of y as the 'next' variable in an infinite list of variables); and
- $\textsc{Ren}^\mu(f(t_1, \ldots, t_k)) = f([t_1]^f_1, \ldots, [t_k]^f_k)$ for evey k-ary symbol f, where given a term $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $[s]^f_i = \textsc{Ren}^\mu(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$.

Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. We define $\mathsf{iDP}(\mathcal{R}, \mu) = \mathsf{iDP}_\mathcal{F}(\mathcal{R}, \mu) \cup \mathsf{iDP}_\mathcal{X}(\mathcal{R}, \mu)$ to be the set of *innermost context-sensitive* dependency pairs (ICS-DPs) where:

$$\mathsf{iDP}_\mathcal{F}(\mathcal{R}, \mu) = \{l^\sharp \to s^\sharp \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s, \textsc{Narr}^\mu(\textsc{Ren}^\mu(s))\}$$
$$\mathsf{iDP}_\mathcal{X}(\mathcal{R}, \mu) = \{l^\sharp \to x \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\}$$

We extend $\mu \in M_{\mathcal{F}}$ into $\mu^\sharp \in M_{\mathcal{F}\cup\mathcal{D}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$. Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. We say that $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ is a *hidden term* if there is a rule $l \to r \in R$ such that $r \rhd_\mu t$. Let $\mathcal{HT}(\mathcal{R}, \mu)$ (or just $\mathcal{HT}$, if $\mathcal{R}$ and μ are clear for the context) be the set of all hidden terms in $(\mathcal{R}, \mu)$. We use $\mathcal{DHT} = \{t \in \mathcal{HT} \mid root(t) \in \mathcal{D}\}$ for the set of hidden terms which are rooted by a *defined* symbol. We also let $\mathcal{NHT}(\mathcal{R}, \mu) = \{t \in \mathcal{DHT} \mid \textsc{Narr}^\mu(\textsc{Ren}^\mu(t))\}$ be the set of *hidden terms* which are rooted by a *defined* symbol, and that, after applying $\textsc{Ren}^\mu$, become *μ-narrowable* (see [4] for further motivation and explanations about these definitions).

Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. An innermost $(\mathcal{P}, \mathcal{R}, \mu)$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution σ satisfying that, for all $i \geq 1$, $\sigma(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$ and :

(i) if $v_i \notin \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i) \hookrightarrow^!_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_{i+1})$, and

(ii) if $v_i \in \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then there is $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\sigma(v_i) \trianglerighteq_\mu s_i$ and $s_i^\sharp \hookrightarrow^!_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_{i+1})$.

As usual, we assume that different occurrences of dependency pairs do not share any variable (renaming substitutions are used if necessary). An innermost $(\mathcal{P}, \mathcal{R}, \mu)$-chain is called *minimal* if for all $i \geq 1$,

(i) if $v_i \notin \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i)$ is innermost $(\mathcal{R}, \mu)$-terminating, and

(ii) if $v_i \in \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $s_i^\sharp$ is innermost $(\mathcal{R}, \mu)$-terminating and $\exists \bar{s}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$ such that $s_i = \sigma(\bar{s}_i)$.

This more abstract notion of chain can be particularized to be used with ICS-DPs, by just taking $\mathcal{P} = \mathsf{iDP}(\mathcal{R}, \mu)$. In the following, the pairs in a CS-TRS $(\mathcal{P}, \mu)$, where $\mathcal{P} = (\mathcal{G}, P)$, are partitioned according to its role in the previous Definition as follows:

$$P_{\mathcal{X}} = \{u \to v \in P \mid v \in \mathcal{V}ar(u) - \mathcal{V}ar^\mu(u)\} \text{ and } P_{\mathcal{G}} = P - P_{\mathcal{X}}$$

Innermost Context-Sensitive Dependency Graph.

Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. The innermost context-sensitive dependency graph consists of the set $\mathsf{iDP}(\mathcal{R}, \mu)$ of innermost context-sensitive dependency pairs and arcs which connect them as follows:

(i) There is an arc from $u \to v \in \mathcal{P}_{\mathcal{G}}$ to $u' \to v' \in \mathcal{P}$ if there are substitutions θ and θ' such that $\theta(v) \hookrightarrow^!_{\mathcal{R},\mu,\mathsf{i}} \theta'(u')$ and $\theta(u), \theta'(u') \in \mathsf{NF}_\mu(\mathcal{R})$.

(ii) There is an arc from $u \to v \in \mathcal{P}_{\mathcal{X}}$ to $u' \to v' \in \mathcal{P}$ if there is $t \in \mathcal{NHT}(\mathcal{R}, \mu)$ and substitutions θ and θ' such that $\theta(t^\sharp) \hookrightarrow^!_{\mathcal{R},\mu,\mathsf{i}} \theta'(u')$ and $\theta'(u') \in \mathsf{NF}_\mu(\mathcal{R})$

In order to approximate the ICS-DG, we have also adapted functions Ren and Cap (used in standard rewriting) to the innermost context-sensitive setting (see [4,5] for details).

Basic usable CS-rules.

Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For any symbol f, let $Rules(\mathcal{R}, \mu, f)$ be the set of rules $l \to r$ defining f and such that the left-hand sides l are argument μ-normalized: $Rules(\mathcal{R}, \mu, f) = \{f(l_1, \ldots, l_k) \to r \in \mathcal{R} \mid \forall i \in \mu(f), l_i \in \mathsf{NF}_\mu(\mathcal{R})\}$. For any term t, the set of *basic* usable CS-rules $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is as follows:

$$\begin{array}{lcl} \mathbf{U}_0(\mathcal{R}, \mu, x) & = & \varnothing \\ \mathbf{U}_0(\mathcal{R}, \mu, f(t_1, \ldots, t_n)) & = & Rules(\mathcal{R}, \mu, f) \cup \bigcup\limits_{i \in \mu(f)} \mathbf{U}_0(\mathcal{R}', \mu, t_i) \cup \bigcup\limits_{l \to r \in Rules(\mathcal{R}, \mu, f)} \mathbf{U}_0(\mathcal{R}', \mu, r) \end{array}$$

where $\mathcal{R}' = \mathcal{R} - Rules(\mathcal{R}, \mu, f)$.

If $\mathcal{P}$ is a TRS, then $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) = \bigcup\limits_{l \to r \in \mathcal{P}} \mathbf{U}_0(\mathcal{R}, \mu, r)$.

4 Simplifying monotonicity requirements for innermost μ-termination

In the innermost setting, matching substitutions are always normalized. For this reason, in an innermost sequence $t_1 \xrightarrow{p_1}_{\mathsf{i}} t_2 \xrightarrow{p_2}_{\mathsf{i}} \cdots \xrightarrow{p_n}_{\mathsf{i}} t_{n+1}$ starting at root position (i.e., $p_1 = \Lambda$), every redex $t_j|_{p_j}$ for $j > 1$ comes from a defined symbol introduced after applying a rule $l_k \to r_k$ in a previous step $k < j$. Hence the set of arguments which are reduced can be handled by looking for defined symbols in right-hand sides of the involved rules $l \to r$.

In [6], Arts and Giesl already noticed that in the treatment of innermost chains, monotonicity requirements for the reduction pairs can be weaker. In [9] Fernández defines the notion of *usable arguments* for a function symbol when proving innermost termination. The idea is that, in innermost sequences, some arguments are not relevant for proving termination.

Example 4.1 Consider the following TRS $\mathcal{R}$:

$$\mathtt{f}(\mathtt{s}(\mathtt{0}), \mathtt{s}(\mathtt{0})) \to \mathtt{f}(x, \mathtt{g}(x)) \qquad\qquad \mathtt{g}(\mathtt{s}(x)) \to \mathtt{g}(x)$$

No innermost sequence starting at root position takes into account the first argument of `f` nor the argument of `g`. The reason is that (instances of) innermost redexes are argument normalized. That means that all variables (e.g. `x`) introduced by the applied rule are normalized and cannot be reduced. Only the second argument `g(x)` of `f` in the right-hand side of the first rule could be innermost reduced after applying it.

Roughly speaking, the usable arguments of a symbol f with respect to a TRS $\mathcal{R}$ are those arguments with a subterm rooted by a defined symbol in some right-hand side of a dependency pair or usable rule.

Definition 4.2 [Usable arguments] [9, Definition 3] Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mathcal{P}$ a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument normalized with respect to $\mathcal{R}$. The set of usable arguments for a function symbol $f \in \mathcal{F}$ with respect to $\mathcal{R}$ and $\mathcal{P}$ is defined as $\mathbf{UA}(f, \mathcal{R}, \mathcal{P}) = \{1 \leq k \leq ar(f) \mid \exists u \to v \in \mathcal{P} \cup \mathbf{U}(\mathcal{R}, \mathcal{P}), \exists p, p' \in \mathcal{P}os(v)$ s.t. $\mathrm{root}(v|_{p'}) = f$, $\mathrm{root}(v|_p) \in \mathcal{D}$, $p'.k \leqslant p$, $u \not\trianglerighteq v|_p\}$.

Considering those usable arguments could be helpful in proofs of innermost termination since they impose weaker monotonicity requirements.

As Fernández noticed, the set of usable arguments can be seen as a replacement map which specifies the arguments to be reduced. According to her results, the μ-termination of a TRS $\mathcal{R}$ implies the innermost termination of $\mathcal{R}$ if $\mu(f)$=$\mathbf{UA}(f, \mathcal{R}, R)$ for all $f \in \mathcal{F}$ where R only contains rules such that all left-hand sides are argument normalized.

Corollary 4.3 *[9, Corollary 11] Let $\mathcal{R}$ be a TRS and $\mu(f) = \mathbf{UA}(f, \mathcal{R}, R')$ for every $f \in \mathcal{F}$ where $R' \subseteq R$ contains all rules $l \to r \in R$ such that l is argument normalized. If $\mathcal{R}$ is μ-terminating, then $\mathcal{R}$ is innermost terminating.*

This observation is very useful since now, all techniques for proving termination of *CSR* can be used for proving innermost termination. Several methods and techniques for proving termination of *CSR* have been developed so far [2,3,14,23].

4.1 Usable arguments for CSR

Following Fernández's ideas, in the innermost context-sensitive setting (for a given replacement map μ) we could relax monotonicity requirements by taking into account that reductions only take place on μ-replacing positions of the right-hand sides of the rules which are rooted by a defined symbol. We adapt Fernández's ideas to *CSR*. In sharp contrast to the unrestricted case, we need to take into account that in innermost *CSR* a redex does not need to be argument normalized. Only argument μ-normalization can be assumed. Thus, non-μ-replacing subterms may contain redexes that can be reduced later on if they come to a replacing position.

Proposition 4.4 *A CS-TRS $(\mathcal{R}, \mu)$ is innermost μ-terminating iff $\mathcal{R}'$ is innermost μ-terminating, where $\mathcal{R}' \subseteq \mathcal{R}$ contains all rules $l \to r \in \mathcal{R}$ such that l is argument μ-normalized.*

Proof. Trivial since the only rules that can be applied in innermost μ-reductions are those whose left-hand sides are argument μ-normalized. □

In the following, we assume that all rules in any CS-TRS $(\mathcal{R}, \mu)$ are argument μ-normalized, i.e., for all rules $l \to r$ in $\mathcal{R}$, l is argument μ-normalized. Proposition 4.4 ensures that this entails no lack of generality regarding our research on innermost termination of *CSR*. The straightforward adaptation of Fernandez's criterion to *CSR* yields the following definition.

Definition 4.5 [Basic usable CS-arguments] Let $(\mathcal{R},\mu) = ((\mathcal{C} \uplus \mathcal{D}, R), \mu)$ be a CS-TRS and $\mathcal{P}$ be a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument μ-normalized. The *basic usable CS-arguments* for a function symbol $f \in \mathcal{F}$ (w.r.t. $\mathcal{R}$ and $\mathcal{P}$) are defined as $\mathbf{UA}_\mu(f,\mathcal{R},\mathcal{P}) = \{i \in \mu(f) \mid \exists u \to v \in \mathcal{P} \cup \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}), \exists p,p' \in \mathcal{P}os^\mu(v)$ s.t. $\mathrm{root}(v|_{p'}) = f$, $root(v|_p) \in \mathcal{D}$, $p'.i \leqslant p$, $u \not\trianglerighteq_\mu v|_p\}$.

Note that the replacement map given by $\mu'(f) = \mathbf{UA}_\mu(f,\mathcal{R},\mathcal{P})$ for all $f \in \mathcal{F}$ is more restrictive than μ, i.e., for all symbols $f \in \mathcal{F}$, $\mu'(f) \subseteq \mu(f)$.

The following proposition is the context-sensitive version of [9, Lemma 5].

Proposition 4.6 *Let $(\mathcal{R},\mu)$ be a CS-TRS and $\mathcal{P}$ be a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument μ-normalized and $\mathcal{P} \cup \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$ is μ-conservative. Let innermost μ-rewriting below the root be $\overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} = (\overset{>\Lambda}{\hookrightarrow} \cap \hookrightarrow_{\mathsf{i}})$. Let $l \to r \in \mathcal{P} \cup \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$ be such that $\sigma(r) \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}\,\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})} t$ for some term t and substitution σ s.t. $\sigma(l)$ is argument μ-normalized . If $t|_p$ is an innermost μ-redex, then for all $p'.k \leqslant p$, we have that $k \in \mathbf{UA}_\mu(root(t|_{p'}),\mathcal{R},\mathcal{P})$.*

Proof. By induction on the length n of the rewriting sequence. If $n = 0$, then $\sigma(r) = t$. Then, since $\sigma(l)$ is argument μ-normalized, it follows that for all $x \in \mathcal{V}ar^\mu(l), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$. Since the rule $l \to r$ is μ-conservative (that is $\mathcal{V}ar^\mu(r) \subseteq \mathcal{V}ar^\mu(l)$), we have that for all $x \in \mathcal{V}ar^\mu(r), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$. It follows that p is a nonvariable (μ-replacing) position of r, i.e. $p \in \mathcal{P}os^\mu_{\mathcal{F}}(r)$. Thus, $\mathrm{root}(r|_p) \in \mathcal{D}$ and the result follows by Definition 4.5.

If $n > 0$, then there is a term s such that $\sigma(r) \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} s$ and $s \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} t$ at some μ-replacing position q. By the induction hypothesis, every μ-replacing position of the term t above, which equal or disjoint to q satisfies the result and we only have to prove it for innermost redexes $t|_p$ s.t. $q < p$, it is say, we have to prove that $k \in \mathbf{UA}_\mu(root(t|_{p'}),\mathcal{R},\mathcal{P})$, for all $q < p'.k \leqslant p$. If $s \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} t$, then $s|_q = \sigma'(l')$ and $t|_q = \sigma'(r')$, for some rule $l' \to r' \in \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$ and substitution σ' s.t. $\sigma'(l')$ is argument μ-normalized. This implies that every innermost redex of $t|_q$ occurs at a position $p'' \in \mathcal{P}os^\mu(r')$ s.t. $root(r'|_{p''}) \in \mathcal{D}$ (since the rule $l' \to r'$ is conservative we have that for all $x \in \mathcal{V}ar^\mu(r'), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$) and $l' \not\trianglerighteq_\mu r'|_{p''}$(otherwise, $\sigma'(l')$ would not be an innermost redex of s. By definition, when $p'' > \Lambda$, $p'.k \leqslant p''$, $k \in \mathbf{UA}_\mu(root(t|_{q.p'}),\mathcal{R},\mathcal{P})$ which is equivalent to what we needed to prove ($k \in \mathbf{UA}_\mu(root(t|_{p'}),\mathcal{R},\mathcal{P})$, for all $q < p'.k \leqslant p$). □

Corollary 4.3 suggests that innermost μ-termination of a TRS $\mathcal{R} = (\mathcal{F},R)$ could be proved as μ'-termination for μ' given by $\mu'(f) = \mathbf{UA}_\mu(f,\mathcal{R},R)$ for all $f \in \mathcal{F}$. This is true for μ-conservative CS-TRSs, as the following theorem shows.

Theorem 4.7 *A μ-conservative CS-TRS $(\mathcal{R},\mu)$ is innermost μ-terminating if $\mathcal{R}$ is μ'-terminating, where for all symbols $f \in \mathcal{F}$, $\mu'(f) = \mathbf{UA}_\mu(f,\mathcal{R},R)$.*

Proof. By contradiction. Assume that $\mathcal{R}$ is not innermost μ-terminating. By the argument of size minimality, there is a infinite innermost μ-rewrite sequence with the first step at position Λ: $s_1 \hookrightarrow_{\mathsf{i}} s_2 \hookrightarrow_{\mathsf{i}} s_3 \hookrightarrow_{\mathsf{i}} \cdots$ (without loss of generality).

By Proposition 4.6 (where we let $\mathcal{P} = \mathcal{R}$), every step $s_j \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} s_{j+1}$ at position p satisfies that $p'.k \leqslant p$, $k \in \mathbf{UA}_\mu(root(s_j|_{p'}), \mathcal{R}, \mathcal{P})$. Therefore, there is an infinite μ'-rewrite sequence of terms $s_1 \hookrightarrow_{\mu'} s_2 \hookrightarrow_{\mu'} \cdots \hookrightarrow_{\mu'} s_n \hookrightarrow_{\mu'} \cdots$ which contradicts the μ'-termination of $\mathcal{R}$.

□

Example 4.8 Consider the TRS $\mathcal{R}$:

$$\mathtt{f}(\mathtt{a},\mathtt{b},x) \to \mathtt{f}(x,x,x)$$
$$\mathtt{c} \to \mathtt{a}$$
$$\mathtt{c} \to \mathtt{b}$$

together with $\mu(\mathtt{f}) = \{1,3\}$. Note that $\mathcal{R}$ is μ-conservative. The set of ICS-DPs consists of the pair $\mathtt{F}(\mathtt{a},\mathtt{b},x) \to \mathtt{F}(x,x,x)$. By using $\mu'(f) = \mathbf{UA}_\mu(f,\mathcal{R},\mathcal{P})$ for every $f \in \mathcal{F}$ we obtain $\mu'(\mathtt{f}) = \varnothing$. The CS-TRS $(\mathcal{R},\mu')$ has no ICS-DP now. Thus we easily conclude the μ'-termination of $\mathcal{R}$ and, by Theorem 4.7, the innermost μ-termination of $\mathcal{R}$.

This fact is important since now, all techniques for proving termination of *CSR* can be used to prove termination of innermost *CSR* for μ-conservative systems. The following example shows that μ-conservativeness cannot be dropped in Theorem 4.7.

Example 4.9 Consider again the TRS $\mathcal{R}$ in Example 4.8 but now together with $\mu(\mathtt{f}) = \{1,2\}$. If we try to apply Theorem 4.7 to prove innermost μ-termination of $\mathcal{R}$, we obtain $\mu'(\mathtt{f}) = \varnothing$ and (as discussed in Example 4.8) we would conclude the innermost μ-termination of $\mathcal{R}$. However, $\mathcal{R}$ is *not* innermost μ-terminating:

$$\underline{\mathtt{f(a,b,c)}} \hookrightarrow_{\mathsf{i}} \mathtt{f(}\underline{\mathtt{c}}\mathtt{,c,c)} \hookrightarrow_{\mathsf{i}} \mathtt{f(a,}\underline{\mathtt{c}}\mathtt{,c)} \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f(a,b,c)}} \hookrightarrow_{\mathsf{i}} \cdots$$

Note that the first rule of $\mathcal{R}$ is not μ-conservative now.

5 Relaxing monotonicity with CS-DPs

Fernández's criterion was also adapted to deal with proofs of termination of rewriting using dependency pairs. We have recently investigated how to prove innermost termination of *CSR* by using (context-sensitive) dependency pairs [5]. Now, we can adapt the use of CS-usable arguments to be applied in proofs of innermost μ-termination with CS-dependency pairs. To give a further step, we do it directly by considering the cycles of the ICS-DG.

Theorem 5.1 *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs, $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$ and $\mathcal{P}$ is μ-conservative be such that $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$ is μ-conservative, and $\mu'(f) = \mathbf{UA}_\mu(f,\mathcal{R},\mathfrak{C})$ for all $f \in \mathcal{F}$. If there is a μ'-reduction pair $(\gtrsim,\sqsupset)$ such that $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}) \subseteq \gtrsim$, $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$, and $\mathcal{P} \cap \sqsupset \neq \varnothing$, then there is no minimal innermost $(\mathcal{R},\mathcal{P},\mu)$-chain.*

Proof. We proceed again by contradiction. Assume that there is a minimal innermost $(\mathcal{R},\mathcal{P},\mu)$-chain:

$$\sigma(u_1) \hookrightarrow_{\mathcal{P},\mu} \sigma(v_1) \overset{>\Lambda}{\longrightarrow}^*_{\mathsf{i}(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu)} \sigma(u_2) \hookrightarrow_{\mathcal{P},\mu} \sigma(v_2) \overset{>\Lambda}{\longrightarrow}^*_{\mathsf{i}(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu)} \sigma(u_3) \hookrightarrow \cdots$$

where all pairs are infinitely often used, and, for all $j \geq 1$, all $u_j \to v_j \in \mathcal{P}$ are μ-conservative and there is a substitution σ such that $\sigma(u_j)$ is argument μ-normalized and $\sigma(v_j)$ is innermost $(\mathcal{R},\mu)$-terminating. By Proposition 4.6, every innermost step in the sequence $\sigma(v_j) \overset{>\Lambda}{\longrightarrow}^*_{\mathsf{i}} \sigma(u_{j+1})$ is performed at a μ'-replacing position by means of a rule in $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$. Since by assumption $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}) \subseteq \gtrsim$, and $\gtrsim$ is stable and μ'-monotonic, $\sigma(v_j) \gtrsim \sigma(u_{j+1})$ holds for all $j > 0$. On the other hand, since $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$ and $\mathcal{P} \cap \sqsupset \neq \varnothing$, the sequence $\sigma(u_1) \gtrsim \cup \sqsupset \sigma(v_1) \gtrsim \sigma(u_2) \cdots$ contains an infinite number of $\sqsupset$ steps. By compatibility of $\gtrsim$ and $\sqsupset$, this contradicts the well-foundedness of $\sqsupset$. □

Theorem 4.7 can be generalized to (certain) non-μ-conservative CS-TRSs thanks to Theorem 5.1 and the results for proving innermost termination of *CSR* in [5]. Now, for a given CS-TRS $(\mathcal{R},\mu)$ that satisfies the conditions of Theorem 5.1, we can prove its innermost μ-termination by relaxing μ-monotonicity requirements for each cycle.

6 Implementation and Experiments

We have implemented the techniques described in the previous sections as part of the termination tool MU-TERM [1,22]. In order to evaluate the techniques which are reported in this paper we have made some benchmarks. We have considered the examples in the 2007 Termination Problem Data Base (TPDB, version 4.0) available through the URL:

http://www.lri.fr/~marche/tpdb

We are going to comment on the results obtained with each improvement.

6.1 *Proving innermost termination of rewriting as termination of* CSR

We have implemented the use of Corollary 4.3 for proving innermost termination of rewriting as termination of *CSR* (this was one of the main results in Fernández's paper). The relevance of this result in practice had not been tested yet. In order to evaluate it we have considered the examples used in the *innermost category* of the 2006 termination Competition [3], which are part of the TPDB (in 2007 the category was not run). There are 69 examples, 66 of them are known to be innermost terminating. With Fernández's criterion (Corollary 4.3) MU-TERM succeeds in 32 examples (success rate of 48.5%). This is acceptable if we think that (except for AProVE, which succeeds in the 100% of the examples) the success rate for all

[3] http://www.lri.fr/~marche/termination-competition/2006

	MU-TERM Fernández	MU-TERM iDPs [6]
YES score	32	39
YES average time	0.03 sec.	0.03 sec.

Table 1
Summary of benchmarks for innermost termination of rewriting

	MU-TERM Fernández cycle-based	MU-TERM Fernández cycle-based (only)
YES score	38	37
YES average time	0.04 sec.	0.79 sec.

Table 2
Summary of benchmarks for innermost termination of rewriting (based on cycles)

other participants in this category is around 20%. However, we have also implemented the use of (standard) dependency pairs for proving innermost termination (according to [6, Theorem 37]) together with the narrowing refinement (we call this tool MU-TERM iDPs) and we are able to prove 39 examples, including all examples solved with Fernández's criterion. Moreover, we have included Fernández's criterion as a technique to be applied when trying to solve a cycle in the innermost termination proof (see [9], Theorem 9). There are two news approximations: when MU-TERM has to solve a cycle, the first version uses Fernández's criterion and if it fails then it tries to solve it in the usual way, that is, without any replacement map (MU-TERM Fernández cycle-based). The second one tries to *force* MU-TERM to solve the cycle with Fernández's criterion, there is no other option (MU-TERM Fernández cycle-based only). In both cases the results obtained are similar. With the previous implementation of MU-TERM (MU-TERM iDPs) we solve 39 examples and with these two configurations we obtain better results than with Corollary 4.3; however, they do not improve the performance of MU-TERM iDPs. The results are summarized in Tables 1 and 2.

Therefore, it seems that using Corollary 4.3 to prove innermost termination of rewriting is not as good idea (at least with the considered set of examples) since we lose some examples due to a too restrictive new replacement map, and the average time is the same. Regarding the application of these ideas to cycles, we obtain better results but no essential improvement since we also lose some examples.

Full details for the benchmarks summarized in Table 1 can be found here:

http://www.dsic.upv.es/~balarcon/prole08/benchmarks/FerInnermost.htm

In the following URL:

http://www.dsic.upv.es/~balarcon/prole08/Innermost/benchmarks.html

more information can be found regarding the benchmarks summarized in Table 2. All this shows that we do not obtain any real improvement over the basic technique of dependency pairs for proving innermost termination at least for the set of considered examples.

6.2 Proving innermost termination of CSR

Although there is no special TPDB category for innermost termination of *CSR*, we have used the TRSs in the *CSR* termination subcategory to test our techniques for proving termination of innermost *CSR* (Theorems 4.7 and 5.1). It contains 90 CS-TRSs.

Since Theorem 4.7 only applies to conservative systems, we restrict the attention to the 27 conservative examples. We solve all of them with an average time of 0.025 seconds (MU-TERMFernández). Further details can be found here:

http://www.dsic.upv.es/~balarcon/prole08/benchmarks/FerICSR.html

On the other hand, we have also implemented the use of Theorem 5.1 to deal with nonconservative systems. We have compared the same configurations explained above: the first one (MU-TERM Fernández cycle-based) tries to solve each μ-conservative cycle (with associated μ-conservative usable rules) by using CS-usable arguments as the new replacement map. If it fails, then the normal configuration of MU-TERM (MU-TERM iCSDPs) is used. The second one only applies CS-usable arguments on cycles when searching for a compatible μ-reduction pair (MU-TERM Fernández cycle-based only). All these versions of MU-TERM succeed over the same 70 examples, the same number of examples that we had already solved using the innermost version of the context-sensitive dependency pairs [5]. The time average rates has no exhibit substantial differences. Further details can be found here:

http://www.dsic.upv.es/~balarcon/prole08/iCSR/benchmarks.html

7 Conclusions and future work

In this work we have shown how to relax monotonicity requirements for proving innermost termination of context-sensitive rewriting. Fernández defined the notion of usable arguments to indicate those arguments that can be eventually reduced in innermost computations. This notion is totally equivalent to fix a replacement map where only those arguments are μ-replacing, thus transforming an innermost termination problem into a context-sensitive termination problem. We have adapted Fernández's approach [9] to be used for proving innermost termination of context-sensitive rewriting (Theorem 4.7). Moreover, since we have recently adapted the use of context-sensitive dependency pairs to deal with innermost termination of *CSR* [5], we have also investigated how to take advantage of it to adapt the result over dependency pairs and reduction pairs of [9] to the context-sensitive setting (Theorem 5.1). We have implemented both, the innermost and the innermost context-sensitive approaches in MU-TERM since the original results had not been implemented or tested before in any termination tool. We have performed some benchmarks that show no real improvement over previous (standard) approaches for proving either innermost termination [6] or innermost termination of *CSR* [5].

One of our main motivations to analyze and continue Fernández's work was the strong connection that she found between *CSR* and innermost rewriting, specifically the possibility of using *CSR* for proving innermost termination of rewriting

that she showed in her paper. Since in 2005 there was no really *powerful* technique for automatically proving termination of *CSR*, her work could not be *tested*, although it is interesting from a theoretical point of view. The recent definition of a context-sensitive version of the dependency pairs approach, though, suggested that some practical benefits could be obtained from her work. After achieving great results in proofs of termination of *CSR* we also developed the innermost version of CS-DPs [5]. The results obtained with this approach were also much better than the existing transformational methods for proving innermost termination of *CSR* [13]. This lead us to think about combining the new CS-DP approach with Fernández's ideas to obtain a good result for proving innermost termination as she stated in her paper. Moreover, she left open the problem of adapting the notion of usable argument to the context-sensitive setting for proving innermost termination of *CSR* which was a nontrivial issue due to the *peculiarities* of context-sensitive reductions as she already noticed. Therefore we have tackled the problem in all its open issues: we have implemented her results in the termination tool MU-TERM and we have also investigated and implemented the extension of her results to *CSR* (even the treatment over cycles that she mentioned). Unfortunately, our practical experience has been very different to what we expected: viewing an innermost termination problem as a context-sensitive termination problem, even with the possibility of using CS-DPs to achieve a proof, does *not* improve the performance of *classical* techniques for proving innermost termination of rewriting (dependency pairs, narrowing, etc [6]). On the other hand, trying to prove innermost termination of CS-TRSs by using a more restrictive replacement map do not offer any improvement over the results obtained by using innermost CS-DPs with the original replacement map [5]. Thus, we can only conclude that, nowadays, apart from the theoretical interest of the approach that combines two *different* strategies, it does not provide a better approach over the existing techniques for proving innermost termination [6] and innermost termination of *CSR* [5]. The other motivation for Fernández's work is that relaxing monotonicity requirements allows the use of non-monotonic orderings for direct proofs (see e.g., [10]). However, as far as we know, there is no tool implementing these orderings in automatic proofs of termination. Some approximations for proving innermost termination using *specific* orderings are presented in [10] but they does not consider the argument positions of function symbols. More research in this field could hopefully clarify whether usable arguments can actually improve the existing methods for proving innermost termination of (context-sensitive) rewriting in some way.

References

[1] Alarcón, B., R. Gutiérrez, J. Iborra and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, volume 188, pages 105–115, 2007.

[2] Alarcón, B., R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In S. Arun-Kumar and N. Garg, editors, *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.

[3] Alarcón, B., R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

[4] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. *Technical Report DSIC II/10/08 (72 pages)*. Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Spain, July 2008.

[5] Alarcón, B., and S. Lucas. Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, LNAI 4720: 73–87, Springer-Verlag, Berlin, 2007.

[6] Arts, T., and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.

[7] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. LNCS 4350, 2007.

[8] Durán, F., S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, 21(1-2):59-88, 2008.

[9] Fernández, M. L. Relaxing monotonicity for innermost termination. Information Processing Letters 93(3):117-123, 2005.

[10] Fernández, M. L., G. Godoy, A. Rubio. Orderings Innermost Termination. In J. Giesl, editor, *Proc. 16th Int. Conf. on Rewriting Techniques and Applications Proc. of RTA'05*, LNCS 3467:17-31, Springer-Verlag, Berlin, 2005.

[11] Futatsugi,K., J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 52-66, ACM Press, 1985.

[12] Futatsugi, K., and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.

[13] Giesl, J., and A. Middeldorp. Innermost termination of context-sensitive rewriting. In M. Ito and M. Toyama, editors, *Proc. of 6th International Conference on Developments in Language Theory, DLT'02*, LNCS 2450:231-244, Springer-Verlag, Berlin, 2003.

[14] Giesl, J., and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379-427, 2004.

[15] Giesl, J., R. Thiemann, P. Schneider-Kamp, and S. Falke Mechanizing and Improving Dependency Pairs *Journal of Automated Reasoning*, 37(3): 155-203. Springer-Verlag, 2006.

[16] Goguen, J.A., T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.

[17] Hirokawa, N., and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199:172-199, 2005.

[18] Lucas, S. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming* 1998(1):1-61, 1998.

[19] Lucas, S. Termination of Rewriting With Strategy Annotations. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, LNAI 2250:669-684, Springer-Verlag, Berlin, 2001.

[20] Lucas, S. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, Proc. of PPDP'01*, pages 82-93, ACM Press, 2001.

[21] Lucas, S. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.

[22] Lucas, S. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of 15h International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at http://www.dsic.upv.es/~slucas/csr/termination/muterm.

[23] Lucas, S. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782-1846, 2006.

18.7 Context-Sensitive Dependency Pairs

7. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. *Information and Computation*, 208:922–968, 2010.

Information and Computation 208 (2010) 922–968

Contents lists available at ScienceDirect

Information and Computation

journal homepage: www.elsevier.com/locate/ic

Context-sensitive dependency pairs ☆

Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas *

ELP Group, DSIC, Universidad Politécnica de Valencia, Spain

ARTICLE INFO

Article history:
Received 11 July 2008
Revised 6 November 2009
Available online 3 April 2010

Keywords:
Dependency pairs
Term rewriting
Program analysis
Termination

ABSTRACT

Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. Although a good number of techniques for proving termination of context-sensitive rewriting (*CSR*) have been proposed so far, the adaptation to *CSR* of the *dependency pair approach*, one of the most powerful techniques for proving termination of rewriting, took some time and was possible only after introducing some new notions like *collapsing dependency pairs*, which are specific for *CSR*. In this paper, we develop the notion of *context-sensitive dependency pair* (CSDP) and show how to use CSDPs in proofs of termination of *CSR*. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for automatically proving termination of *CSR*.

© 2010 Elsevier Inc. All rights reserved.

1. Introduction

Most computational systems whose operational principle is based on reducing expressions can be described and analyzed by using notions and techniques from the abstract framework of term rewriting systems (TRSs [11,61]). Such computational systems (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) often incorporate a predefined reduction strategy that is used to break down the nondeterminism that is inherent to reduction relations. Eventually, this can raise problems, as each kind of strategy only behaves properly for particular classes of programs (i.e., it is normalizing, optimal, etc.). For this reason, the designers of programming languages have developed mechanisms to give the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to arguments of symbols) have been used in programming languages such as Clean [57], Haskell [41], Lisp [54], Maude [14], OBJ2 [23], OBJ3 [36], CafeOBJ [24], etc. to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become 'more eager' and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become 'more lazy', thus (hopefully) avoiding nontermination. Termination is one of the most interesting practical problems in computation and software engineering. A program or computational system is said to be *terminating* if it does not lead to any infinite computation for any possible call or input data. Ensuring termination is often a prerequisite for essential program properties like correctness. Messages reporting (a never-ending) "processing", "waiting for an answer", or even "abnormal termination" (which are often raised during the execution of software applications) usually correspond to nonterminating computations arising from bugs in the program.

Context-sensitive rewriting (*CSR* [44,46]) is a restriction of rewriting that has proved useful in investigating some of the aforementioned programming languages, see e.g., [13,16,17,31,45,52]. In *CSR*, the restriction of the rewriting computations is

☆ This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under Grants TIN 2007-68093-C02 and HA 2006-0007. Beatriz Alarcón was partially supported by the Spanish MEC/MICINN under FPU Grant AP2005-3399. Raúl Gutiérrez was partially supported by the Spanish MEC/MICINN Grant TIN 2004-7943-C04-02.

* Corresponding author.
Email addresses: balarcon@dsic.upv.es (B. Alarcón), rgutierrez@dsic.upv.es (R. Gutiérrez), slucas@dsic.upv.es (S. Lucas).
URLs: http://www.dsic.upv.es/~balarcon (B. Alarcón), http://www.dsic.upv.es/~rgutierrez (R. Gutiérrez), http://www.dsic.upv.es/~slucas (S. Lucas).

0890-5401/$ - see front matter © 2010 Elsevier Inc. All rights reserved.
doi:10.1016/j.ic.2010.03.003

$$
\begin{aligned}
\texttt{evenNs} &\to \texttt{cons}(\texttt{0}, \texttt{incr}(\texttt{oddNs}))\\
\texttt{oddNs} &\to \texttt{incr}(\texttt{evenNs})\\
\texttt{incr}(\texttt{cons}(x, xs)) &\to \texttt{cons}(\texttt{s}(x), \texttt{incr}(xs))\\
\texttt{take}(\texttt{0}, xs) &\to \texttt{nil}\\
\texttt{take}(\texttt{s}(n), \texttt{cons}(x, xs)) &\to \texttt{consF}(x, \texttt{take}(n, xs))\\
\texttt{zip}(\texttt{nil}, xs) &\to \texttt{nil}\\
\texttt{zip}(xs, \texttt{nil}) &\to \texttt{nil}\\
\texttt{zip}(\texttt{cons}(x, xs), \texttt{cons}(y, ys)) &\to \texttt{cons}(\texttt{frac}(x, y), \texttt{zip}(xs, ys))\\
\texttt{tail}(\texttt{cons}(x, xs)) &\to xs\\
\texttt{rep2}(\texttt{nil}) &\to \texttt{nil}\\
\texttt{rep2}(\texttt{cons}(x, xs)) &\to \texttt{cons}(x, \texttt{cons}(x, \texttt{rep2}(xs)))\\
\texttt{add}(\texttt{0}, n) &\to n\\
\texttt{add}(\texttt{s}(n), m) &\to \texttt{s}(\texttt{add}(n, m))\\
\texttt{prod}(\texttt{0}, n) &\to \texttt{0}\\
\texttt{prod}(\texttt{s}(n), m) &\to \texttt{add}(m, \texttt{prod}(n, m))\\
\texttt{prodFrac}(\texttt{frac}(x, y), \texttt{frac}(z, t)) &\to \texttt{frac}(\texttt{prod}(x, z), \texttt{prod}(y, t))\\
\texttt{prodOfFracs}(\texttt{nil}) &\to \texttt{frac}(\texttt{s}(\texttt{0}), \texttt{s}(\texttt{0}))\\
\texttt{prodOfFracs}(\texttt{consF}(p, ps)) &\to \texttt{prodFrac}(p, \texttt{prodOfFracs}(ps))\\
\texttt{halfPi}(n) &\to \texttt{prodOfFracs}(\texttt{take}(n, \texttt{zip}(\texttt{rep2}(\texttt{tail}(\texttt{evenNs})), \texttt{tail}(\texttt{rep2}(\texttt{oddNs})))))
\end{aligned}
$$

Fig. 1. Computing Wallis' approximation to $\frac{\pi}{2}$.

first imposed on the *arguments* of function symbols f in the signature $\mathcal{F}$. A *signature* is a set of function symbols $f_1, \ldots, f_n, \ldots$ together with an *arity* function $ar: \mathcal{F} \to \mathbb{N}$ that establishes the number of 'arguments' associated to each symbol. A *replacement map* is a mapping $\mu: \mathcal{F} \to \wp(\mathbb{N})$ that satisfies $\mu(f) \subseteq \{1, \ldots, ar(f)\}$, for each symbol f in the signature $\mathcal{F}$ [44]. It specifies the argument positions where rewriting is allowed. In *CSR*, we only rewrite μ-replacing subterms: every term t (as a whole) is μ-replacing by definition; and t_i (as well as all its μ-replacing subterms) is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$.

Example 1. The TRS $\mathcal{R}$ in Fig. 1 can be used to compute approximations to $\frac{\pi}{2}$ by using Wallis' product: $\frac{\pi}{2} = lim_{n\to\infty} \frac{2}{1}\frac{2}{3}\frac{4}{3}\frac{4}{5}\cdots\frac{2n}{2n-1}\frac{2n}{2n+1}$. In $\mathcal{R}$, function symbols `0` and `s` are used to represent natural numbers in Peano's notation; we also have the usual arithmetic operations `add`ition and `prod`uct. Symbols `cons` and `nil` are the standard list constructors which are then used to build (possibly infinite) lists of natural numbers like `evenNs` (the infinite list of even numbers) and `oddNs` (the infinite list of odd numbers). Function `incr` increases all the elements of a list in one unit through the application of `s`. The function `zip` merges a pair of lists into a list of fractions, and `tail` returns the elements of a list after removing the first one. The function `take` can be used to obtain the components of a finite approximation to $\frac{\pi}{2}$ which we multiply with `prodOfFracs`. Note the explicit use of `consF` for building *finite* lists of fractions of natural numbers by means of `take`, thus ensuring that the product of their elements computed by `prodOfFracs` is well-defined. A call $\texttt{halfPi}(\texttt{s}^n(\texttt{0}))$ for some positive number $n > 0$ will return the desired approximation. Since $\mathcal{R}$ is *nonterminating* (due to the first two rules), we should be careful when choosing the rewrite steps that will be issued to obtain an approximation.

With *CSR* we can achieve a *terminating behaviour* for this system. Consider the replacement map μ given by:

$$\mu(\texttt{cons}) = \{1\} \text{ and } \mu(f) = \{1, \ldots, ar(f)\} \text{ for all } f \in \mathcal{F} - \{\texttt{cons}\}$$

where $\mu(\texttt{cons}) = \{1\}$ disallows reductions on the *list* part of the list constructor `cons`, thus making a kind of *lazy evaluation* of lists possible. Furthermore, the replacement restrictions imposed by the replacement map μ are *not* an obstacle to obtaining the desired approximations: the repeated application of context-sensitive rewriting steps to an expression $\texttt{halfPi}(\texttt{s}^n(\texttt{0}))$ will obtain (disregarding the particular choice of such steps) an expression $\texttt{frac}(\texttt{s}^p(\texttt{0}), \texttt{s}^q(\texttt{0}))$ representing the approximation $\frac{p}{q}$ to $\frac{\pi}{2}$, which is obtained by taking the first n terms in Wallis' formula (this follows from [44, Theorem 11]).

1.1. Termination of context-sensitive rewriting

Several methods have been developed for proving termination of *CSR* under a replacement map μ for a given TRS $\mathcal{R}$ (i.e., for proving the μ-*termination* of $\mathcal{R}$). Termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and in the analysis of programming languages [8,16,17,19,21,31,46,50,59]. The development of methods and techniques for automatically proving termination is, therefore, one of the most interesting and challenging problems

$$\mathtt{ADD}(\mathtt{s}(n), m) \to \mathtt{ADD}(n, m) \quad (1)$$

$$\mathtt{EVENNS} \to \mathtt{INCR}(\mathtt{oddNs}) \quad (2)$$

$$\mathtt{EVENNS} \to \mathtt{ODDNS} \quad (3)$$

$$\mathtt{HALFPI}(n) \to \mathtt{EVENNS} \quad (4)$$

$$\mathtt{HALFPI}(n) \to \mathtt{ODDNS} \quad (5)$$

$$\mathtt{HALFPI}(n) \to \mathtt{PRODOFFRACS}(\mathtt{take}(n, \mathtt{zip}(\mathtt{rep2}(\mathtt{tail}(\mathtt{evenNs})), \mathtt{tail}(\mathtt{rep2}(\mathtt{oddNs}))))) \quad (6)$$

$$\mathtt{HALFPI}(n) \to \mathtt{REP2}(\mathtt{oddNs}) \quad (7)$$

$$\mathtt{HALFPI}(n) \to \mathtt{REP2}(\mathtt{tail}(\mathtt{evenNs})) \quad (8)$$

$$\mathtt{HALFPI}(n) \to \mathtt{TAIL}(\mathtt{evenNs}) \quad (9)$$

$$\mathtt{HALFPI}(n) \to \mathtt{TAIL}(\mathtt{rep2}(\mathtt{oddNs})) \quad (10)$$

$$\mathtt{HALFPI}(n) \to \mathtt{TAKE}(n, \mathtt{zip}(\mathtt{rep2}(\mathtt{tail}(\mathtt{evenNs})), \mathtt{tail}(\mathtt{rep2}(\mathtt{oddNs})))) \quad (11)$$

$$\mathtt{HALFPI}(n) \to \mathtt{ZIP}(\mathtt{rep2}(\mathtt{tail}(\mathtt{evenNs})), \mathtt{tail}(\mathtt{rep2}(\mathtt{oddNs}))) \quad (12)$$

$$\mathtt{INCR}(\mathtt{cons}(x, xs)) \to \mathtt{INCR}(xs) \quad (13)$$

$$\mathtt{ODDNS} \to \mathtt{EVENNS} \quad (14)$$

$$\mathtt{ODDNS} \to \mathtt{INCR}(\mathtt{evenNs}) \quad (15)$$

$$\mathtt{PROD}(\mathtt{s}(n), m) \to \mathtt{ADD}(m, \mathtt{prod}(n, m)) \quad (16)$$

$$\mathtt{PROD}(\mathtt{s}(n), m) \to \mathtt{PROD}(n, m) \quad (17)$$

$$\mathtt{PRODFRAC}(\mathtt{frac}(x, y), \mathtt{frac}(z, t)) \to \mathtt{PROD}(x, z) \quad (18)$$

$$\mathtt{PRODFRAC}(\mathtt{frac}(x, y), \mathtt{frac}(z, t)) \to \mathtt{PROD}(y, t) \quad (19)$$

$$\mathtt{PRODOFFRACS}(\mathtt{consF}(p, ps)) \to \mathtt{PRODFRAC}(p, \mathtt{prodOfFracs}(ps)) \quad (20)$$

$$\mathtt{PRODOFFRACS}(\mathtt{consF}(p, ps)) \to \mathtt{PRODOFFRACS}(ps) \quad (21)$$

$$\mathtt{REP2}(\mathtt{cons}(x, xs)) \to \mathtt{REP2}(xs) \quad (22)$$

$$\mathtt{TAKE}(\mathtt{s}(n), \mathtt{cons}(x, xs)) \to \mathtt{TAKE}(n, xs) \quad (23)$$

$$\mathtt{ZIP}(\mathtt{cons}(x, xs), \mathtt{cons}(y, ys)) \to \mathtt{ZIP}(xs, ys) \quad (24)$$

Fig. 2. Dependency pairs for the TRS in Example 1.

when dealing with *CSR*. Furthermore, with *CSR*, we can *achieve* a terminating behavior with nonterminating TRSs by pruning (all) infinite rewrite sequences as shown in Example 1. Examples of tools that are able to automatically prove termination of *CSR* are AProVE [32], Jambox [18], MU-TERM [2,47], and VMTL [60].

In the 90s, a number of transformations that permit termination of *CSR* to be treated as a standard termination problem were developed (see [31,50] for recent surveys). Polynomial orderings and the context-sensitive version of the recursive path ordering were also investigated [12,26,48,49]. In [3], we adapted the *dependency pair method* [10,25,38], which is a very powerful technique for proving termination of rewriting, to *CSR*. In this paper, we develop and improve the original notions in [3] to incorporate recent improvements introduced by the dependency pair framework [33,35], and we obtain a powerful and modern framework that improves the current state-of-the-art of methods that can be used to automatically prove termination of *CSR*. Our tool MU-TERM implements the methods and techniques described in this paper.

1.2. Dependency pairs for context-sensitive rewriting

A TRS $\mathcal{R}$ is terminating if there is no infinite rewrite sequence starting from any term. With regard to proofs of termination of rewriting, the dependency pair technique focuses on the following idea: the rules that are really able to produce such infinite sequences are those rules $l \to r$ such that r contains some *defined* symbol[1] g. Intuitively, we can think of these rules as representing some possible (direct or indirect) recursive calls. Such recursion paths associated to each rule $l \to r$ are represented as new rules $u \to v$, where $u = f^\sharp(l_1, \ldots, l_k)$ if $l = f(l_1, \ldots, l_k)$, and where $v = g^\sharp(s_1, \ldots, s_m)$ if $s = g(s_1, \ldots, s_m)$ is a subterm of r and g is a defined symbol. The notation $f^\sharp$ for a given symbol f means that f is *marked*. In

[1] A symbol $g \in \mathcal{F}$ is defined in $\mathcal{R}$ if there is a rule in $\mathcal{R}$ whose left-hand side is of the form $g(l_1, \ldots, l_k)$.

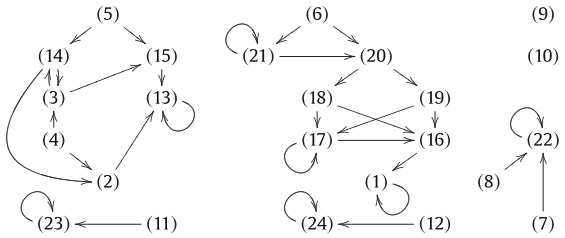


Fig. 3. Dependency graph for the TRS in Example 1.

practice, we often capitalize f and use F instead of $f^\sharp$ in our examples. For this reason, the dependency pair technique starts by considering a new TRS $\mathsf{DP}(\mathcal{R})$ that contains all these new rules for each $l \to r \in \mathcal{R}$. For instance, according to [10], the set $\mathsf{DP}(\mathcal{R})$ of dependency pairs for $\mathcal{R}$ in Example 1 consists of the rules in Fig. 2. The rules in $\mathcal{R}$ and the rules in $\mathsf{DP}(\mathcal{R})$ determine the so-called *dependency chains* whose finiteness or infiniteness characterize termination or nontermination of $\mathcal{R}$ [10]. A chain of dependency pairs is a sequence $u_i \to v_i$ of dependency pairs together with a substitution σ such that $\sigma(v_i)$ rewrites to $\sigma(u_{i+1})$ for all $i \geqslant 1$. The dependency pairs can be presented as a *dependency graph*, where the infinite chains are represented by the *cycles* in the graph. For instance, the dependency graph that corresponds to the TRS $\mathcal{R}$ in Example 1 is depicted in Fig. 3. The cycle consisting of nodes (3) and (14) witnesses the *nontermination* of $\mathcal{R}$.

In general, these intuitions are valid for *CSR*: the subterms s of the right-hand sides r of the rules $l \to r$ which are considered to build the *context-sensitive dependency pairs* $l^\sharp \to s^\sharp$ must be μ*-replacing* terms now.

Example 2. Consider $\mathcal{R}$ and μ as in Example 1. Only the dependency pairs (1), (4)-(12), (14)-(21), and (23) in Fig. 2 are also context-sensitive dependency pairs.

The following example shows the need for dependency pairs of a *new kind*.

Example 3. Consider the following TRS $\mathcal{R}$:

$$\mathtt{a} \to \mathtt{c}(\mathtt{f}(\mathtt{a})) \qquad\qquad \mathtt{f}(\mathtt{c}(x)) \to x$$

together with $\mu(\mathtt{c}) = \varnothing$ and $\mu(\mathtt{f}) = \{1\}$. No μ-replacing subterm s in the right-hand sides of the rules is rooted by a defined symbol. Thus, there is no 'regular' dependency pair (in particular $\mathtt{A} \to \mathtt{A}$ is *dismissed* due to $\mu(\mathtt{c}) = \varnothing$). If no other dependency pair is considered, we could wrongly conclude that $\mathcal{R}$ is μ-terminating, which is not true:

$$\mathtt{f}(\underline{\mathtt{a}}) \hookrightarrow_\mu \underline{\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a})))} \hookrightarrow_\mu \mathtt{f}(\underline{\mathtt{a}}) \hookrightarrow_\mu \cdots$$

Indeed, we must add the following *collapsing* dependency pair:

$$\mathtt{F}(\mathtt{c}(x)) \to x.$$

Since the right-hand side is a variable, this would not be allowed in Arts and Giesl's approach [10].

Collapsing pairs are essential in our approach. They express that infinite context-sensitive rewrite sequences can involve not only the kind of recursion that is represented by the *usual* dependency pairs but also a new kind of recursion that is *hidden* inside the nonreplacing (or *frozen*) parts of the terms involved in the infinite sequence. The *activation* of such *delayed* recursions is due to the presence of *migrating* variables within a rule $l \to r$ which is used in the sequence. Migrating variables are those that are not replacing in the left-hand side l but that *become* replacing in the right-hand side r.

Example 4 (*Continuing Example 2*)**.** The following *collapsing* pairs are *context-sensitive dependency pairs* for the CS-TRS in Example 1:

$$\mathtt{TAIL}(\mathtt{cons}(x, xs)) \to xs \tag{25}$$

$$\mathtt{TAKE}(\mathtt{s}(n), \mathtt{cons}(x, xs)) \to xs \tag{26}$$

Note that variable xs is μ-replacing in the right-hand sides of the rules $\mathtt{tail}(\mathtt{cons}(x, xs)) \to xs$ and $\mathtt{take}(\mathtt{s}(n), \mathtt{cons}(x, xs)) \to \mathtt{consF}(x, \mathtt{take}(n, xs))$ but it is *non-*μ*-replacing* in the corresponding left-hand sides.

1.3. Plan of the paper

We have argued that termination of *CSR* is an interesting and challenging topic of research with a good number of practical applications. The results, techniques, and tools that derive from our work can be of interest to a sufficiently wide audience. The material in this paper will be more familiar, however, to those specialists who are interested in termination (in general) and in *how* to prove termination of *CSR* in particular. Throughout the paper, however, we made a serious effort to provide sufficient intuition and informal descriptions for our main definitions and results.

After Section 2, the paper is structured in three main parts:

1. Section 3 provides appropriate notions of *minimal* non-μ-terminating terms and introduces the main properties of such terms. We introduce the notion of *hidden term* and investigate the structure of infinite context-sensitive rewrite sequences starting from minimal non-μ-terminating terms. This analysis is essential in order to provide an appropriate definition of context-sensitive dependency pair and the related notions of chains, graphs, etc.
2. We define the notions of *context-sensitive dependency pair* and *context-sensitive chain of pairs* and show how to use them to *characterize* termination of *CSR*. Sections 4 and 5 introduce the general framework to compute and use context-sensitive dependency pairs to prove termination of *CSR*. The introduction of dependency pairs of a new kind (the *collapsing* dependency pairs, as in Example 3) leads to a notion of context-sensitive dependency *chain*, which is quite different from the standard one. In Section 6, we prove that our *context-sensitive dependency pair approach* fully characterizes termination of *CSR*.
3. We describe a suitable *framework* for dealing with proofs of termination of *CSR* by using these results. Section 7 adapts the *dependency pair framework* [33,35] to *CSR* by defining appropriate notions of *CS problem* and *CS processor* that rely on the results obtained in the second part of the paper. Section 8 introduces the notion of *context-sensitive (dependency) graph* and the associated CS processor. Section 9 describes CS processors for removing or transforming collapsing pairs. Section 10 investigates the use of term orderings in processors. Section 11 adapts Hirokawa and Middeldorp's *subterm criterion* [38]. Section 12 adapts *narrowing transformation* of pairs in [35].

Experiments are reported in Section 13. Sections 14 and 15 discuss related work. Section 16 concludes.

2. Preliminaries

This section collects a number of definitions and notations about term rewriting. More details and missing notions can be found in [11,58,61].

Let A be a set and $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$ be a binary relation on A. We denote the transitive closure of R by R^+ and its reflexive and transitive closure by R^*. We say that R is *terminating* (*strongly normalizing*) if there is no infinite sequence $a_1\ \mathsf{R}\ a_2\ \mathsf{R}\ a_3 \cdots$. A reflexive and transitive relation R is a quasi-ordering.

Given relations R and R' over the same set A, we define its *composition* $\mathsf{R} \circ \mathsf{R}'$ as follows: for all $a, b \in \mathsf{A}$, $a\ (\mathsf{R} \circ \mathsf{R}')\ b$ if there is $c \in \mathsf{A}$ such that $a\ \mathsf{R}\ c$ and $c\ \mathsf{R}'\ b$.

2.1. Signatures, terms, and positions

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. $\mathcal{V}ar(t)$ is the set of variables occurring in a term t. A term t is *ground* if it contains no variable (i.e., $\mathcal{V}ar(t) = \emptyset$). A term is said to be *linear* if it has no multiple occurrences of a single variable.

Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. We denote the empty chain by Λ. Given positions p, q, we denote their concatenation as $p \cdot q$. Positions are ordered by the standard prefix ordering: $p \leqslant q$ if $\exists q'$ such that $q = p \cdot q'$. If p is a position, and Q is a set of positions, then $p \cdot Q = \{p \cdot q \mid q \in Q\}$. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of nonvariable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$, and $t[s]_p$ is the term t with the subterm at position p replaced by s.

We write $s \trianglerighteq t$, read *t is a subterm of s*, if $t = s|_p$ for some $p \in \mathcal{P}os(s)$ and $s \triangleright t$ if $s \trianglerighteq t$ and $s \neq t$. We write $s \not\trianglerighteq t$ and $s \not\triangleright t$ for the negation of the corresponding properties. The symbol labeling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with a 'hole' $\Box$ (a fresh constant symbol). We write $C[\ ]_p$ to denote that there is a (usually single) hole $\Box$ at position p of C. Generally, we write $C[\ ]$ to denote an arbitrary context and make the position of the hole explicit only if necessary. $C[\ ] = \Box$ is called the *empty* context.

2.2. Substitutions, renamings, and unifiers

A substitution is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$. Denote as ε the 'identity' substitution: $\varepsilon(x) = x$ for all $x \in \mathcal{X}$. The set $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ is called the *domain* of σ.

Remark 1. We do *not* impose that the domain of the substitutions be finite. This is usual practice in the dependency pair approach, where a single substitution is used to instantiate an infinite number of variables coming from renamed versions of the dependency pairs (see below).

A *renaming* is an injective substitution ρ such that $\rho(x) \in \mathcal{X}$ for all $x \in \mathcal{X}$. A substitution σ such that $\sigma(s) = \sigma(t)$ for two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is called a *unifier* of s and t; we also say that s and t unify (with substitution σ). If two terms s and t unify, then there is a unique *most general unifier* σ (up to renaming of variables) such that for every other unifier τ, there is a substitution θ such that $\theta \circ \sigma = \tau$.

A relation $\mathsf{R} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X})$ on terms is *stable* if, for all terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and substitutions σ, we have $\sigma(s)\ \mathsf{R}\ \sigma(t)$ whenever $s\ \mathsf{R}\ t$.

2.3. Rewrite systems and term rewriting

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l, and the right-hand side (*rhs*) is r. A rewrite rule $l \to r$ is said to be *collapsing* if $r \in \mathcal{X}$. A *Term Rewriting System* (TRS) is a pair $\mathcal{R} = (\mathcal{F}, R)$, where R is a set of rewrite rules. We often use $\varnothing$ to denote TRSs whose set of rules R is empty. Given TRSs $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{R}' = (\mathcal{F}', R')$, we let $\mathcal{R} \cup \mathcal{R}'$ be the TRS $(\mathcal{F} \cup \mathcal{F}', R \cup R')$. An instance $\sigma(l)$ of a *lhs* l of a rule is called a *redex*. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$ (called *constructors*) and symbols $f \in \mathcal{D}$ (called *defined functions*), where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Example 5. Consider again the TRS in Example 1. The symbols `evenNs`, `oddNs`, `incr`, `take`, `zip`, `tail`, `rep2`, `add`, `prod`, `prodFrac`, `prodOfFracs`, and `halfPi` are defined. Symbols `s`, `0`, `cons`, `consF`, `nil`, and `frac` are constructors.

We often write $l \to r \in \mathcal{R}$ instead of $l \to r \in R$ to express that the rule $l \to r$ is a rule of $\mathcal{R}$. A term $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to t (at position p), written $s \xrightarrow{p}_{\mathcal{R}} t$ (or just $s \to_{\mathcal{R}} t$, or $s \to t$), if $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(s)$ and substitution σ. We write $s \xrightarrow{>p}_{\mathcal{R}} t$ if $s \xrightarrow{q}_{\mathcal{R}} t$ for some $q > p$. A TRS $\mathcal{R}$ is terminating if its one step rewrite relation $\to_{\mathcal{R}}$ is terminating.

2.4. Context-sensitive rewriting

A mapping $\mu : \mathcal{F} \to \wp(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if for all symbols $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ [44]. Let $M_{\mathcal{F}}$ be the set of all $\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F}, R)$). Let $\mu_\top$ be the replacement map given by $\mu_\top(f) = \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}$ (i.e., no replacement restrictions are specified).

A binary relation R on terms is μ-monotonic if, for all $f \in \mathcal{F}$, $i \in \mu(f)$, and $s, t, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $f(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_k)\ \mathsf{R}\ f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_k)$ whenever $s\ \mathsf{R}\ t$. If R is $\mu_\top$-monotonic, we just say that R is *monotonic*.

The set of *μ-replacing positions* $\mathcal{P}os^\mu(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$ if $t \in \mathcal{X}$, and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^\mu(t|_i)$ if $t \notin \mathcal{X}$. Note that $\mathcal{P}os^\mu(t)$ (as $\mathcal{P}os(t)$) is *prefix closed*. When no replacement map is made explicit, the μ-replacing positions are often called *active*; and the non-μ-replacing ones are often called *frozen*. The following results about *CSR* are often used without any explicit mention.

Proposition 1 [44]. *Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $p = q \cdot q' \in \mathcal{P}os(t)$. Then $p \in \mathcal{P}os^\mu(t)$ iff $q \in \mathcal{P}os^\mu(t) \wedge q' \in \mathcal{P}os^\mu(t|_q)$.*

The chain of symbols lying on positions above/on $p \in \mathcal{P}os(t)$ is $prefix_t(\Lambda) = root(t)$, $prefix_t(i \cdot p) = root(t).prefix_{t|_i}(p)$. The strict prefix sprefix is $\mathrm{sprefix}_t(\Lambda) = \Lambda$, $\mathrm{sprefix}_t(p \cdot i) = prefix_t(p)$, i.e., the last symbol in $prefix_t(p \cdot i)$ is *removed*. Although $\mathrm{sprefix}_t(p)$ is a sequence, when the ordering of symbols in $\mathrm{sprefix}_t(p)$ does not matter, we often use the standard set-theoretic notation (e.g., inclusion as in $\mathrm{sprefix}_t(p) \subseteq \mathcal{F}$) with the obvious meaning.

Proposition 2 [44]. *If $p \in \mathcal{P}os(t) \cap \mathcal{P}os(s)$ and $sprefix_t(p) = sprefix_s(p)$, then $p \in \mathcal{P}os^\mu(t) \Leftrightarrow p \in \mathcal{P}os^\mu(s)$.*

The μ-replacing subterm relation $\trianglerighteq_\mu$ is given by $s \trianglerighteq_\mu t$ if there is $p \in \mathcal{P}os^\mu(s)$ such that $t = s|_p$. We write $s \triangleright_\mu t$ if $s \trianglerighteq_\mu t$ and $s \neq t$. We write $s \triangleright_{\not\mu} t$ to denote that t is a non-μ-replacing (hence strict) subterm of s: $s \triangleright_{\not\mu} t$ if there is $p \in \mathcal{P}os(s) - \mathcal{P}os^\mu(s)$ such that $t = s|_p$. The set of *μ-replacing* variables of a term t, i.e., variables occurring at some μ-replacing position in t, is $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid t \trianglerighteq_\mu x\}$. The set of non-$\mu$-replacing variables of t, i.e., variables occurring at some non-μ-replacing position in t, is $\mathcal{V}ar^{\not\mu}(t) = \{x \in \mathcal{V}ar(t) \mid t \triangleright_{\not\mu} x\}$. Note that $\mathcal{V}ar^\mu(t)$ and $\mathcal{V}ar^{\not\mu}(t)$ do not need to be disjoint (when t is not linear).

A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a CS-TRS. In *context-sensitive rewriting*, we (only) contract μ-*replacing* redexes: s μ-rewrites to t, written $s \overset{p}{\hookrightarrow}_{\mathcal{R},\mu} t$ (or $s \hookrightarrow_{\mathcal{R},\mu} t$, $s \hookrightarrow_{\mu} t$ and even $s \hookrightarrow t$), if $s \overset{p}{\rightarrow}_{\mathcal{R}} t$ and $p \in \mathcal{P}os^{\mu}(s)$.

Example 6. Consider $\mathcal{R}$ and μ as in Example 1. Then, we have:

$$\underline{\texttt{evenNs}} \hookrightarrow_{\mu} \texttt{cons}(\texttt{0}, \texttt{incr}(\underline{\texttt{oddNs}})) \not\hookrightarrow_{\mu} \texttt{cons}(\texttt{0}, \texttt{incr}(\texttt{incr}(\texttt{evenNs})))$$

Since the second argument of `cons` is not μ-replacing, we have $2 \notin \mathcal{P}os^{\mu}(\texttt{cons}(\texttt{0}, \texttt{incr}(\texttt{oddNs})))$. Thus, redex `oddNs` cannot be μ-rewritten.

A term t is μ-terminating (or $(\mathcal{R}, \mu)$-terminating, if we want an explicit reference to the involved TRS $\mathcal{R}$) if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \cdots \hookrightarrow_{\mathcal{R},\mu} t_n \hookrightarrow_{\mathcal{R},\mu} \cdots$ starting from t. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_{\mathcal{R},\mu}$ is terminating.

A term s μ-*narrows* to a term t (written $s \rightsquigarrow_{\mathcal{R},\mu,\theta} t$), if there is a nonvariable μ-replacing position $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(s)$ and a rule $l \to r$ in $\mathcal{R}$ (sharing no variable with s) such that $s|_p$ and l unify with the most general unifier θ and $t = \theta(s[r]_p)$. The following definition is used in Section 10.2.

Definition 1 [26]**.** Let $\mathcal{F}$ be a signature and $\mu \in M_{\mathcal{F}}$. The μ-replacing projection TRS $\mathcal{E}mb^{\mu}(\mathcal{F})$ consists of the following rules:

$$\{f(x_1, \ldots, x_k) \to x_i \mid f \in \mathcal{F}, i \in \mu(f)\}$$

3. Minimal non-μ-terminating terms and infinite μ-rewrite sequences

Given a TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$, the *minimal* nonterminating terms associated to $\mathcal{R}$ are nonterminating terms t whose proper subterms u (i.e., $t \rhd u$) are terminating; $\mathcal{T}_{\infty}$ is the set of minimal nonterminating terms associated to $\mathcal{R}$ [38,40]. Minimal nonterminating terms have two important properties:

1. Every nonterminating term s contains a minimal nonterminating term $t \in \mathcal{T}_{\infty}$ (i.e., $s \trianglerighteq t$).
2. Minimal nonterminating terms t are always rooted by a *defined* symbol $f \in \mathcal{D}$: $\forall t \in \mathcal{T}_{\infty}$, $root(t) \in \mathcal{D}$.

As discussed in [38], considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t \in \mathcal{T}_{\infty}$ can be helpful to come to the notion of dependency pair [10]. Such sequences proceed as follows:

Proposition 3 [38, Lemma 1]**.** *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS. For all $t \in \mathcal{T}_{\infty}$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{T}_{\infty}$ such that $root(u) \in \mathcal{D}$, $t \overset{>\Lambda}{\longrightarrow}{}^{*} \sigma(l) \overset{\Lambda}{\rightarrow} \sigma(r) \trianglerighteq u$, and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.*

In the following, we show how to generalize these notions and results to *CSR*.

3.1. Minimal non-μ-terminating terms

Before starting our discussion about (minimal) non-μ-terminating terms, we provide an obvious auxiliary result about μ-terminating terms.[2]

Lemma 1. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If s is μ-terminating, then:*

1. *If $s \trianglerighteq_{\mu} t$, then t is μ-terminating.*
2. *If $s \hookrightarrow^{*}_{\mathcal{R},\mu} t$, then t is μ-terminating.*

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ and a replacement map $\mu \in M_{\mathcal{F}}$, maybe the simplest extension to *CSR* of the notion of minimal term for unrestricted rewriting (i.e., $\mathcal{T}_{\infty}$), is the following: let $\mathcal{T}_{\infty,\mu}$ be a set of minimal non-μ-terminating terms in the following sense: t belongs to $\mathcal{T}_{\infty,\mu}$ if t is non-μ-terminating and every strict subterm u (i.e., $t \rhd u$) is μ-terminating. It is obvious that $root(t) \in \mathcal{D}$ for all $t \in \mathcal{T}_{\infty,\mu}$. We also have the following:

Lemma 2. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If s is not μ-terminating, then there is a subterm t of s ($s \trianglerighteq t$) such that $t \in \mathcal{T}_{\infty,\mu}$.*

[2] For the sake of readability, the missing proofs of the technical results in this section have been moved to Appendix A.

Unfortunately, there can be non-μ-terminating terms having no μ-*replacing* subterm in $\mathcal{T}_{\infty,\mu}$.

Example 7. Consider the CS-TRS $(\mathcal{R}, \mu)$ in Example 3 and $s = \mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a})))$. Note that s is not μ-terminating, but $s \notin \mathcal{T}_{\infty,\mu}$ because $\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a}))) \rhd \mathtt{f}(\mathtt{a})$ and $\mathtt{f}(\mathtt{a})$ is not μ-terminating. Note that $\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a}))) \rhd_{\not\mu} \mathtt{f}(\mathtt{a})$. The only μ-replacing strict subterm of s is $\mathtt{c}(\mathtt{f}(\mathtt{a}))$, which is μ-terminating, i.e., $\mathtt{c}(\mathtt{f}(\mathtt{a})) \notin \mathcal{T}_{\infty,\mu}$.

Therefore, minimal non-μ-terminating terms are not the most natural ones because they could occur at non-μ-replacing positions, where no μ-rewriting step is possible. Thus, this simple notion would not lead to an appropriate generalization of Proposition 3 to *CSR*. There is a suitable generalization of Proposition 3 to *CSR* (see Proposition 5) based on the following notion.

Definition 2 (*Minimal non-μ-terminating term*)**.** Let $\mathcal{M}_{\infty,\mu}$ be a set of minimal non-μ-terminating terms in the following sense: t belongs to $\mathcal{M}_{\infty,\mu}$ if t is non-μ-terminating and every strict μ-*replacing* subterm t' of t (i.e., $t \rhd_\mu t'$) is μ-terminating.

Note that $\mathcal{T}_{\infty,\mu} \subseteq \mathcal{M}_{\infty,\mu}$. In the following, we often say that terms in $\mathcal{T}_{\infty,\mu}$ are *strongly minimal* non-μ-terminating; we use them in Section 3.4. Now, we have the following:

Lemma 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_\mathcal{F}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If s is not μ-terminating, then there is a μ-replacing subterm t of s such that $t \in \mathcal{M}_{\infty,\mu}$.*

Obviously, if $t \in \mathcal{M}_{\infty,\mu}$, then $root(t)$ is a defined symbol. Since μ-terminating terms are preserved under μ-rewriting (Lemma 1), it follows that $\mathcal{M}_{\infty,\mu}$ is preserved under *inner* μ-rewritings in the following sense.

Lemma 4. *Let $\mathcal{R}$ be a TRS, $\mu \in M_\mathcal{R}$, and $t \in \mathcal{M}_{\infty,\mu}$. If $t \overset{>\Lambda}{\hookrightarrow}{}^* u$ and u is non-μ-terminating, then $u \in \mathcal{M}_{\infty,\mu}$.*

Lemma 4 does *not* hold for $\mathcal{T}_{\infty,\mu}$: consider the CS-TRS $(\mathcal{R}, \mu)$ in Example 3. Note that $\mathtt{f}(\mathtt{a}) \in \mathcal{T}_{\infty,\mu}$ and $\mathtt{f}(\underline{\mathtt{a}}) \overset{>\Lambda}{\hookrightarrow} \mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a})))$. Although $\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a})))$ is not μ-terminating, $\mathtt{f}(\mathtt{c}(\mathtt{f}(\mathtt{a}))) \notin \mathcal{T}_{\infty,\mu}$, as shown in Example 7.

3.2. Hidden terms in minimal μ-rewrite sequences

Given a CS-TRS $(\mathcal{R}, \mu)$, the *hidden terms* are nonvariable terms occurring on some frozen position in the right-hand side of some rule of $\mathcal{R}$. As we show in the next section, they play an important role in infinite minimal μ-rewrite sequences associated to $\mathcal{R}$.

Definition 3 (*Hidden symbols and terms*)**.** Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. We say that $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ is a *hidden term* if there is a rule $l \to r \in R$ such that $r \rhd_{\not\mu} t$. Let $\mathcal{HT}(\mathcal{R}, \mu)$ (or just $\mathcal{HT}$, if no confusion arises) be the set of all hidden terms in $(\mathcal{R}, \mu)$. We say that $f \in \mathcal{F}$ is a *hidden symbol* if it occurs in a hidden term. Let $\mathcal{H}(\mathcal{R}, \mu)$ (or just $\mathcal{H}$) be the set of all hidden symbols in $(\mathcal{R}, \mu)$.

In the following, we also use $\mathcal{DHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT}(\mathcal{R}, \mu) \mid root(t) \in \mathcal{D}\}$ for the set of hidden terms which are rooted by a *defined* symbol.

Example 8. For $\mathcal{R}$ and μ as in Example 1, the maximal hidden terms are $\mathtt{incr}(\mathtt{oddNs})$, $\mathtt{incr}(x)$, $\mathtt{zip}(xs, ys)$, and $\mathtt{cons}(x, \mathtt{rep2}(xs))$. The hidden symbols are $\mathtt{incr}$, $\mathtt{oddNs}$, $\mathtt{incr}$, $\mathtt{zip}$, $\mathtt{cons}$, and $\mathtt{rep2}$. Finally, $\mathcal{DHT}(\mathcal{R}, \mu) = \{\mathtt{oddNs}, \mathtt{incr}(\mathtt{oddNs}), \mathtt{incr}(x), \mathtt{zip}(xs, ys), \mathtt{rep2}(xs)\}$.

The following lemma says that frozen subterms t in the contractum $\sigma(r)$ of a redex $\sigma(l)$ that do *not* contain t are (at least partly) 'introduced' by a hidden term in the right-hand side r of the involved rule $l \to r$.

Lemma 5. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution. If there is a rule $l \to r \in R$ such that $\sigma(l) \not\rhd t$ and $\sigma(r) \rhd_{\not\mu} t$, then there is no $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$. Furthermore, there is a term $t' \in \mathcal{HT}$ such that $r \rhd_{\not\mu} t'$ and $\sigma(t') = t$.*

The following lemma establishes that minimal non-μ-terminating and non-μ-replacing subterms that occur in a μ-rewrite sequence involving only minimal terms come directly from the first term in the sequence or are instances of a hidden term.

Lemma 6. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let A be a μ-rewrite sequence $t_1 \hookrightarrow t_2 \hookrightarrow \cdots \hookrightarrow t_n$ with $t_i \in \mathcal{M}_{\infty,\mu}$ for all i, $1 \leqslant i \leqslant n$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_1 \not\trianglerighteq t$ and $t_n \rhd_{\not\mu} t$, then $t = \sigma(s)$ for some $s \in \mathcal{DHT}$ and substitution σ.*

We use the previous results to investigate infinite sequences that combine μ-rewriting steps on minimal non-μ-terminating terms and the extraction of such subterms as μ-replacing subterms of (instances of) right-hand sides of the rules.

Proposition 4. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Consider a finite or infinite sequence of the form $t_1 \overset{\Lambda}{\hookrightarrow} s_1 \trianglerighteq_\mu t_2' \overset{>\Lambda}{\hookrightarrow}{}^* t_2 \overset{\Lambda}{\hookrightarrow} s_2 \trianglerighteq_\mu t_3' \overset{>\Lambda}{\hookrightarrow}{}^* t_3 \cdots$ with $t_i, t_i' \in \mathcal{M}_{\infty,\mu}$ for all $i \geq 1$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_i \rhd_{\not\mu} t$ for some $i \geq 1$, then $t_1 \rhd_{\not\mu} t$ or $t = \sigma(s)$ for some $s \in \mathcal{DHT}$ and substitution σ.*

3.3. Infinite μ-rewrite sequences starting from minimal terms

The following proposition establishes that, given a minimal non-μ-terminating term $t \in \mathcal{M}_{\infty,\mu}$, there are only two ways for an infinite μ-rewrite sequence to proceed. The first one is by using 'visible' parts of the rules that correspond to μ-replacing nonvariable subterms in the right-hand sides that are rooted by a defined symbol. The second one is by showing up 'hidden' non-μ-terminating subterms that are activated by *migrating* variables in a rule $l \to r$, i.e., variables $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$ that are *not* μ-replacing in the left-hand side l but become μ-replacing in the right-hand side r.

Proposition 5. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Then, for all $t \in \mathcal{M}_{\infty,\mu}$, there exist $l \to r \in \mathcal{R}$, a substitution σ, and a term $u \in \mathcal{M}_{\infty,\mu}$ such that $t \overset{>\Lambda}{\hookrightarrow}{}^* \sigma(l) \overset{\Lambda}{\hookrightarrow} \sigma(r) \trianglerighteq_\mu u$ and either*

1. *there is a nonvariable μ-replacing subterm s of r, $r \trianglerighteq_\mu s$, such that $u = \sigma(s)$, or*
2. *there is $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$ such that $\sigma(x) \trianglerighteq_\mu u$.*

Proof. Consider an infinite μ-rewrite sequence starting from t. By definition of $\mathcal{M}_{\infty,\mu}$, all proper μ-replacing subterms of t are μ-terminating. Therefore, t has an inner reduction to an instance $\sigma(l)$ of the left-hand side of a rule $l \to r$ of $\mathcal{R}$: $t \overset{>\Lambda}{\hookrightarrow}{}^* \sigma(l) \overset{\Lambda}{\hookrightarrow} \sigma(r)$ and $\sigma(r)$ is not μ-terminating. Thus, we can write $t = f(t_1, \ldots, t_k)$ and $\sigma(l) = f(l_1, \ldots, l_k)$ for some k-ary defined symbol f, and $t_i \hookrightarrow^* \sigma(l_i)$ for all i, $1 \leqslant i \leqslant k$. Since all t_i are μ-terminating for $i \in \mu(f)$, by Lemma 1, $\sigma(l_i)$ and all its μ-replacing subterms are also μ-terminating. In particular, $\sigma(y)$ is μ-terminating for all μ-replacing variables y in l: $y \in \mathcal{V}ar^\mu(l)$. Since $\sigma(r)$ is non-μ-terminating, by Lemma 3, it contains a μ-replacing subterm $u \in \mathcal{M}_{\infty,\mu}$: $\sigma(r) \trianglerighteq_\mu u$, i.e., there is a position $p \in \mathcal{P}os^\mu(\sigma(r))$ such that $\sigma(r)|_p = u$. We consider two cases:

1. If $p \in \mathcal{P}os_{\mathcal{F}}(r)$ is a nonvariable position of r, then there is a μ-replacing nonvariable subterm s of r (i.e., $p \in \mathcal{P}os^\mu_{\mathcal{F}}(r)$ and $s = r|_p \notin \mathcal{X}$), such that $u = \sigma(s)$.
2. If $p \notin \mathcal{P}os_{\mathcal{F}}(r)$, then there is a μ-replacing variable position $q \in \mathcal{P}os^\mu(r) \cap \mathcal{P}os_{\mathcal{X}}(r)$ such that $q \leqslant p$. Let $x \in \mathcal{V}ar^\mu(r)$ be such that $r|_q = x$. Then, $\sigma(x) \trianglerighteq_\mu u$, and $\sigma(x)$ is not μ-terminating: since $u \in \mathcal{M}_{\infty,\mu}$ is not μ-terminating, by Lemma 1, $\sigma(x)$ is not μ-terminating. Since $\sigma(y)$ is μ-terminating for all $y \in \mathcal{V}ar^\mu(l)$, we conclude that $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$. □

Proposition 5 entails the following result, which establishes some properties of infinite sequences starting from minimal non-μ-terminating terms.

Corollary 1. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For all $t \in \mathcal{M}_{\infty,\mu}$, there is an infinite sequence*

$$t \overset{>\Lambda}{\hookrightarrow}{}^* \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}{}^* \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}{}^* \cdots$$

where, for all $i \geqslant 1$, $l_i \to r_i \in \mathcal{R}$ are rewrite rules, σ_i are substitutions, and terms $t_i \in \mathcal{M}_{\infty,\mu}$ are minimal non-μ-terminating terms such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable subterm s_i such that $r_i \trianglerighteq_\mu s_i$, or*
2. *$\sigma_i(x_i) \trianglerighteq_\mu t_i$ for some $x_i \in \mathcal{V}ar^\mu(r_i) - \mathcal{V}ar^\mu(l_i)$.*

Remark 2. The $(\hookrightarrow_\mu \cup \trianglerighteq_\mu)$-sequence in Corollary 1 can be easily viewed as an infinite μ-rewrite sequence by just introducing appropriate contexts $C_i[\ ]_{p_i}$ with μ-replacing holes: since $\sigma_i(r_i) \trianglerighteq_\mu t_i$, there is $p_i \in \mathcal{P}os^\mu(\sigma_i(r_i))$ such that $\sigma_i(r_i) = \sigma_i(r_i)[t_i]_{p_i}$; just take $C_i[\ ]_{p_i} = \sigma(r_i)[\Box]_{p_i}$. Hence:

$$t \hookrightarrow^* \sigma_1(l_1) \hookrightarrow C_1[t_1]_{p_1} \hookrightarrow^* C_1[\sigma_2(l_2)]_{p_1} \hookrightarrow C_1[C_2[t_2]_{p_2}]_{p_1} \hookrightarrow^* \cdots$$

Note that, e.g., $p_1 \cdot p_2 \in \mathcal{P}os^\mu(C_1[C_2[t_2]_{p_2}]_{p_1})$ (use Proposition 1).

3.4. Infinite μ-rewrite sequences starting from strongly minimal terms

In the following, we consider a function $\textsc{Ren}^\mu$ which *independently* renames all *occurrences* of μ-replacing variables within a term t by using new fresh variables that are not in $\mathcal{V}ar(t)$:

- $\textsc{Ren}^\mu(x) = y$ if x is a variable, where y is intended to be a fresh new variable that has not yet been used;
- $\textsc{Ren}^\mu(f(t_1, \ldots, t_k)) = f([t_1]^f_1, \ldots, [t_k]^f_k)$ for evey k-ary symbol f, where given a term $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $[s]^f_i = \textsc{Ren}^\mu(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$.

Note that $\textsc{Ren}^\mu(t)$ keeps variables at non-μ-replacing positions untouched. Note also that $\textsc{Ren}^\mu$ is *not* a substitution: it replaces the $n(x)$ different μ-replacing occurrences of the *same* variable x by *different* variables $x_1, \ldots, x_{n(x)}$. Clearly, $t = \theta(\textsc{Ren}^\mu(t))$ for some substitution θ which just identifies the variables introduced by $\textsc{Ren}^\mu$ (i.e., $\theta(x_i) = x$ for all $1 \leqslant i \leqslant n(x)$). The use of $\textsc{Ren}^\mu$ together with μ-narrowability yields a necessary condition for reducibility of terms under some instantiations which is used in our development.

Proposition 6. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution. If $\sigma(t) \overset{>\Lambda}{\hookrightarrow}{}^* \sigma(l)$ for some (possibly renamed) rule $l \to r \in \mathcal{R}$, then $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

Proof. We can write the sequence from $\sigma(t)$ to $\sigma(l)$ as follows: $\sigma(t) = t_1 \overset{>\Lambda}{\hookrightarrow} t_2 \overset{>\Lambda}{\hookrightarrow} \cdots \overset{>\Lambda}{\hookrightarrow} t_m = \sigma(l)$ for some $m \geqslant 1$. We proceed by induction on m.

1. If $m = 1$, then $\sigma(t) = \sigma(l)$. Since $t \notin \mathcal{X}$, t is μ-narrowable (at the root position) using the rule $l \to r$. Since $t = \theta(\textsc{Ren}^\mu(t))$ for some substitution θ, we have $\sigma(t) = \sigma(\theta(\textsc{Ren}^\mu(t))) = \sigma(l)$. Since we can assume that the new variables instantiated by θ are not in l, we have $\sigma(\theta(l)) = \sigma(l)$. Thus, $\textsc{Ren}^\mu(t)$ and l unify with mgu $\sigma \circ \theta$. Since $t \notin \mathcal{X}$, implies that $\textsc{Ren}^\mu(t) \notin \mathcal{X}$, $\textsc{Ren}^\mu(t)$ is μ-narrowable at the root position using the same rule $l \to r$.
2. If $m > 1$, then we have $t_1 \overset{>\Lambda}{\hookrightarrow} t_2 \overset{>\Lambda}{\hookrightarrow}{}^* \sigma(l)$. We consider two cases according to the position $p \in \mathcal{P}os^\mu(t_1)$ where the μ-rewrite step $t_1 \overset{>\Lambda}{\hookrightarrow} t_2$ is performed (note that $t_1 = \sigma(t)$ by assumption).
   (a) If $p \in \mathcal{P}os^\mu_\mathcal{F}(t)$, then there is a rule $l' \to r'$ and a substitution θ such that $\sigma(t)|_p = \sigma(t|_p) = \theta(l')$. Again, we have $\sigma(t|_p) = \sigma(l')$, i.e., t is μ-narrowable at position p using rule $l' \to r'$ and (reasoning as above), we conclude that $\textsc{Ren}^\mu(t)$ is μ-narrowable.
   (b) If $p \notin \mathcal{P}os^\mu_\mathcal{F}(t)$, then there is a μ-replacing variable position $q \in \mathcal{P}os^\mu_\mathcal{X}(t)$ of t such that $t|_q = x \in \mathcal{V}ar^\mu(t)$, $q \leqslant p$ and $\sigma(x) \hookrightarrow_\mu t_2|_q$. Therefore, $t_1 = \sigma(t[x]_q) = \sigma(t)[\sigma(x)]_q$ and $t_2 = \sigma(t)[t_2|_q]_q = \sigma'(t')$ for a term $t' = t[y]_q$ where y is a new fresh variable $y \notin \mathcal{V}ar(t)$ and a substitution σ' given by $\sigma'(y) = t_2|_q$ and $\sigma'(z) = \sigma(z)$ for all $z \in \mathcal{V}ar(t)$ (including x). Clearly,

$$\sigma'(t') = \sigma'(t[y]_q) = \sigma'(t)[\sigma'(y)]_q = \sigma(t)[t_2|_q]_q = t_2.$$

   By the induction hypothesis, $\textsc{Ren}^\mu(t')$ is μ-narrowable. Since t and t' only differ in a single variable, we can assume that $\textsc{Ren}^\mu(t') = \textsc{Ren}^\mu(t)$. Thus, we conclude that $\textsc{Ren}^\mu(t)$ is μ-narrowable as well. □

Corollary 2. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution such that $\sigma(t) \in \mathcal{M}_{\infty,\mu}$. Then, $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

Proof. By Proposition 5, there is a rule $l \to r$ and a substitution σ such that $\sigma(t) \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathcal{R},\mu} \sigma(l)$ (since we can assume that variables in l and variables in t are disjoint, we can apply the same substitution σ to t and l without any problem). By Proposition 6, the conclusion follows. □

In the following, we write $\textsc{Narr}^\mu_\mathcal{R}(t)$ (or just $\textsc{Narr}^\mu(t)$) to indicate that t is μ-narrowable with respect to the (intended) TRS $\mathcal{R}$. We also let

$$\mathcal{NHT}(\mathcal{R}, \mu) = \left\{ t \in \mathcal{DHT}(\mathcal{R}, \mu) \mid \textsc{Narr}^\mu_\mathcal{R}(\textsc{Ren}^\mu(t)) \right\}$$

be the set of *hidden terms* that are rooted by a *defined* symbol, and that after applying $\textsc{Ren}^\mu$ become μ-*narrowable*.

Example 9. Since all terms $t \in \mathcal{DHT}(\mathcal{R}, \mu)$ for $\mathcal{R}$ and μ as in Example 8 are μ-narrowable (even without applying REN^{μ}), we have $\mathcal{NHT}(\mathcal{R}, \mu) = \mathcal{DHT}(\mathcal{R}, \mu)$.

As a consequence of the previous results, we have the following main result, which we use later.

Theorem 1. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For all $t \in \mathcal{T}_{\infty,\mu}$, there is an infinite sequence*

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}{}^{*} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow} \sigma_1(r_1) \trianglerighteq_{\mu} t_1 \overset{>\Lambda}{\hookrightarrow}{}^{*} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow} \sigma_2(r_2) \trianglerighteq_{\mu} t_2 \overset{>\Lambda}{\hookrightarrow}{}^{*} \cdots$$

where, for all $i \geqslant 1$, $l_i \to r_i \in \mathcal{R}$ are rewrite rules, σ_i are substitutions, and terms $t_i \in \mathcal{M}_{\infty,\mu}$ are minimal non-μ-terminating terms such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable term s_i such that $r_i \trianglerighteq_{\mu} s_i$, or*
2. *$\sigma_i(x_i) \trianglerighteq_{\mu} t_i$ for some $x_i \in \mathcal{V}ar^{\mu}(r_i) - \mathcal{V}ar^{\mu}(l_i)$ and $t_i = \theta_i(t_i')$ for some $t_i' \in \mathcal{NHT}$ and substitution θ_i.*

Proof. Since $\mathcal{T}_{\infty,\mu} \subseteq \mathcal{M}_{\infty,\mu}$, by Corollary 1, we have a sequence

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}{}^{*} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow} \sigma_1(r_1) \trianglerighteq_{\mu} t_1 \overset{>\Lambda}{\hookrightarrow}{}^{*} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow} \sigma_2(r_2) \trianglerighteq_{\mu} t_2 \overset{>\Lambda}{\hookrightarrow}{}^{*} \cdots$$

where, for all $i \geqslant 1$, $l_i \to r_i \in \mathcal{R}$, σ_i are substitutions, $t_i \in \mathcal{M}_{\infty,\mu}$, and either (1) $t_i = \sigma_i(s_i)$ for some nonvariable term s_i such that $r_i \trianglerighteq_{\mu} s_i$ or (2) $\sigma_i(x_i) \trianglerighteq_{\mu} t_i$ for some $x_i \in \mathcal{V}ar^{\mu}(r_i) - \mathcal{V}ar^{\mu}(l_i)$ (and hence $\sigma(l_i) \rhd_{\not\mu} t_i$ and $\sigma(r_i) \trianglerighteq_{\mu} t_i$ as well). We only need to prove that terms t_i are instances of hidden terms in $\mathcal{NHT}$ whenever (2) holds. By Proposition 4, for all such terms t_i, we have that either (*A*) $\sigma_1(l_1) \rhd_{\not\mu} t_i$ or (*B*) $t_i = \theta_i(t_i')$ for some $t_i' \in \mathcal{DHT}$ and substitution θ_i. In case (*B*), we just consider Corollary 2, which ensures that $t_i' \in \mathcal{NHT}$. In case (*A*), since $t \overset{>\Lambda}{\hookrightarrow}{}^{*} \sigma_1(l_1)$ and $\sigma_1(l_1)$ is not μ-terminating, by Lemma 4, all terms u_j in the μ-rewrite sequence

$$t = u_1 \overset{>\Lambda}{\hookrightarrow} u_2 \overset{>\Lambda}{\hookrightarrow} \cdots \overset{>\Lambda}{\hookrightarrow} u_m = \sigma_1(l_1)$$

belong to $\mathcal{M}_{\infty,\mu}$: $u_j \in \mathcal{M}_{\infty,\mu}$ for all j, $1 \leqslant j \leqslant m$. Since $t \in \mathcal{T}_{\infty,\mu}$, all its strict subterms (disregarding their μ-replacing character) are μ-terminating. Since t_i is not μ-terminating, $t \not\rhd t_i$. By Lemma 6, $t_i = \theta_i(t_i')$ for some $t_i' \in \mathcal{DHT}$ and substitution θ_i. By Corollary 2, $t_i' \in \mathcal{NHT}$. □

4. Context-sensitive dependency pairs

By Lemma 2 every non-μ-terminating term s_0 contains a strongly minimal subterm $t \in \mathcal{T}_{\infty,\mu}$ which, by Theorem 1, starts an infinite μ-rewrite sequence. In such a sequence, a number of μ-rewriting steps *below the root* of t are performed. Then a rule $l \to r$ is applied at the *topmost* position of the obtained reduct. According to Proposition 5, the application of such a rule either

1. *introduces* a new minimal non-μ-terminating subterm u having a prefix s which is a nonvariable μ-replacing subterm of r. By Corollary 2, $\text{REN}^{\mu}(s)$ is μ-narrowable. Otherwise,
2. *takes* a minimal non-μ-terminating and non-μ-replacing subterm u and
   (a) brings it up to an *active* position by means of the binding $\sigma(x)$ for some *migrating variable* x in $l \to r$.
   (b) At this point, we know that u, which is rooted by a defined symbol due to $u \in \mathcal{M}_{\infty,\mu}$, is an instance of a hidden term $u' \in \mathcal{NHT}$.

Afterwards, further *inner* μ-rewritings on u lead to a matching with the left-hand-side l' of a new rule $l' \to r'$ and everything starts again. This process is abstracted in the definition of *context-sensitive dependency pairs* and in the definition of chain below.

Given a signature $\mathcal{F}$ and $f \in \mathcal{F}$, we let $f^{\sharp}$ be a new fresh symbol (often called *tuple* symbol or DP-symbol) associated to a symbol f [10]. Let $\mathcal{F}^{\sharp}$ be the set of tuple symbols associated to symbols in $\mathcal{F}$. As usual, for $t = f(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we write $t^{\sharp}$ to denote the *marked* term $f^{\sharp}(t_1, \ldots, t_k)$. Conversely, given a marked term $t = f^{\sharp}(t_1, \ldots, t_k)$, where $t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, we write $t^{\flat}$ to denote the term $f(t_1, \ldots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.

Definition 4 (*Context-sensitive dependency pairs*)**.** Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. Let $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ be the set of *context-sensitive* dependency pairs (CSDPs) where:

$$\mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu) = \left\{ l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s, \textsc{Narr}^\mu(\textsc{Ren}^\mu(s)) \right\}$$
$$\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) = \left\{ l^\sharp \to x \mid l \to r \in R, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l) \right\}$$

We extend $\mu \in M_{\mathcal{F}}$ into $\mu^\sharp \in M_{\mathcal{F}\cup\mathcal{D}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$.

The CSDPs $u \to v \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ in Definition 4, consisting of collapsing rules only, are called the *collapsing* CSDPs.

Remark 3. The notion of CSDP in Definition 4 differs from the standard definition of dependency pair [10,35] in two additional requirements:

1. As in [38], which follows Dershowitz's proposal in [15], we require that subterms s of the right-hand sides r of the rules $l \to r$ which are considered to build the dependency pairs $l^\sharp \to s^\sharp$ are not subterms of the left-hand side (i.e., $l \not\trianglerighteq_\mu s$).
2. As in [53], we require μ-narrowability of $\textsc{Ren}^\mu(s)$: $\textsc{Narr}^\mu(\textsc{Ren}^\mu(s))$.

But the crucial difference, which is specific for context-sensitive rewriting, is the introduction and use of *collapsing* dependency pairs.

A rule $l \to r$ of a TRS $\mathcal{R}$ is μ-conservative if $\mathcal{V}ar^\mu(r) \subseteq \mathcal{V}ar^\mu(l)$, i.e., there is no migrating variable; $\mathcal{R}$ is μ-conservative if all its rules are μ-conservative (see [43,50]). The following fact is obvious from Definition 4.

Proposition 7. *If $\mathcal{R}$ is a μ-conservative TRS, then* $\mathsf{DP}(\mathcal{R},\mu) = \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$.

Therefore, in order to deal with μ-conservative TRSs $\mathcal{R}$ we only need to consider the 'classical' dependency pairs in $\mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$.

$$\texttt{ADD}(\texttt{s}(n), m) \to \texttt{ADD}(n, m) \quad (1)$$
$$\texttt{HALFPI}(n) \to \texttt{EVENNS} \quad (4)$$
$$\texttt{HALFPI}(n) \to \texttt{ODDNS} \quad (5)$$
$$\texttt{HALFPI}(n) \to \texttt{PRODOFFRACS}(\texttt{take}(n, \texttt{zip}(\texttt{rep2}(\texttt{tail}(\texttt{evenNs})), \texttt{tail}(\texttt{rep2}(\texttt{oddNs}))))) \quad (6)$$
$$\texttt{HALFPI}(n) \to \texttt{REP2}(\texttt{oddNs}) \quad (7)$$
$$\texttt{HALFPI}(n) \to \texttt{REP2}(\texttt{tail}(\texttt{evenNs})) \quad (8)$$
$$\texttt{HALFPI}(n) \to \texttt{TAIL}(\texttt{evenNs}) \quad (9)$$
$$\texttt{HALFPI}(n) \to \texttt{TAIL}(\texttt{rep2}(\texttt{oddNs})) \quad (10)$$
$$\texttt{HALFPI}(n) \to \texttt{TAKE}(n, \texttt{zip}(\texttt{rep2}(\texttt{tail}(\texttt{evenNs})), \texttt{tail}(\texttt{rep2}(\texttt{oddNs})))) \quad (11)$$
$$\texttt{HALFPI}(n) \to \texttt{ZIP}(\texttt{rep2}(\texttt{tail}(\texttt{evenNs})), \texttt{tail}(\texttt{rep2}(\texttt{oddNs}))) \quad (12)$$
$$\texttt{ODDNS} \to \texttt{EVENNS} \quad (14)$$
$$\texttt{ODDNS} \to \texttt{INCR}(\texttt{evenNs}) \quad (15)$$
$$\texttt{PROD}(\texttt{s}(n), m) \to \texttt{ADD}(m, \texttt{prod}(n, m)) \quad (16)$$
$$\texttt{PROD}(\texttt{s}(n), m) \to \texttt{PROD}(n, m) \quad (17)$$
$$\texttt{PRODFRAC}(\texttt{frac}(x, y), \texttt{frac}(z, t)) \to \texttt{PROD}(x, z) \quad (18)$$
$$\texttt{PRODFRAC}(\texttt{frac}(x, y), \texttt{frac}(z, t)) \to \texttt{PROD}(y, t) \quad (19)$$
$$\texttt{PRODOFFRACS}(\texttt{consF}(p, ps)) \to \texttt{PRODFRAC}(p, \texttt{prodOfFracs}(ps)) \quad (20)$$
$$\texttt{PRODOFFRACS}(\texttt{consF}(p, ps)) \to \texttt{PRODOFFRACS}(ps) \quad (21)$$
$$\texttt{TAKE}(\texttt{s}(n), \texttt{cons}(x, xs)) \to \texttt{TAKE}(n, xs) \quad (23)$$
$$\texttt{TAIL}(\texttt{cons}(x, xs)) \to xs \quad (25)$$
$$\texttt{TAKE}(\texttt{s}(n), \texttt{cons}(x, xs)) \to xs \quad (26)$$

Fig. 4. Context-sensitive dependency pairs for the CS-TRS in Example 1.

Example 10. Consider the following TRS $\mathcal{R}$:

$$\mathtt{g}(x) \to \mathtt{h}(x) \qquad \mathtt{h}(\mathtt{d}) \to \mathtt{g}(\mathtt{c}) \qquad \mathtt{c} \to \mathtt{d}$$

together with $\mu(\mathtt{g}) = \mu(\mathtt{h}) = \varnothing$ [63, Example 1]. Note that $\mathcal{R}$ is μ-conservative. $\mathsf{DP}(\mathcal{R}, \mu)$ consists of the following (noncollapsing) CSDPs:

$$\mathtt{G}(x) \to \mathtt{H}(x) \qquad \mathtt{H}(\mathtt{d}) \to \mathtt{G}(\mathtt{c})$$

with $\mu^\sharp(\mathtt{G}) = \mu^\sharp(\mathtt{H}) = \varnothing$.

If the TRS $\mathcal{R}$ contains non-μ-conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

Example 11. As discussed in Examples 2 and 4, for the CS-TRS $(\mathcal{R}, \mu)$ in Example 1, we have the CSDPs in Fig. 4.

5. Chains of CSDPs

An essential property of the dependency pair method is that it provides a *characterization* of termination of TRSs $\mathcal{R}$ as the absence of infinite (minimal) *chains of dependency pairs* [10,35]. As we prove in Section 6, this is also true for *CSR* when CSDPs are considered. First, we have to introduce a suitable notion of chain that can be used with CSDPs. As in the DP-framework [33,35], where the origin of *pairs* does not matter, we use another TRS $\mathcal{P}$ together with $\mathcal{R}$ to build the chains. Once this more abstract notion of chain is introduced, it can be particularized to be used with CSDPs, by just taking $\mathcal{P} = \mathsf{DP}(\mathcal{R}, \mu)$.

Definition 5 (*Chain of pairs – minimal chain*)**.** Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. A $(\mathcal{P}, \mathcal{R}, \mu)$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$ satisfying that, for all $i \geqslant 1$:

1. if $v_i \notin \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$, and
2. if $v_i \in \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i) = C_i[s_i]_{p_i}$ for some s_i and $C_i[\,]_{p_i}$ such that $p_i \in \mathcal{P}os^\mu(C_i[\,]_{p_i})$, $\mathrm{sprefix}_{C_i[\,]_{p_i}}(p_i) \subseteq \mathcal{F}$, and $s_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$.

As usual, we assume that different occurrences of pairs do not share any variable (renaming substitutions are used if necessary). A $(\mathcal{P}, \mathcal{R}, \mu)$-chain is called *minimal* if for all $i \geqslant 1$,

1. if $v_i \notin \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $\sigma(v_i)$ is $(\mathcal{R}, \mu)$-terminating, and
2. if $v_i \in \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$, then $s_i^\sharp$ is $(\mathcal{R}, \mu)$-terminating and $\exists \bar{s}_i \in \mathcal{NT}(\mathcal{R}, \mu)$ such that $s_i = \sigma(\bar{s}_i)$.

Note that the condition $v_i \in \mathcal{V}ar(u_i) - \mathcal{V}ar^\mu(u_i)$ in Definition 5 implies that v_i is a variable. Furthermore, v_i is a *migrating variable* in the rule $u_i \to v_i$.

Remark 4 (*Conventions about* $\mathcal{P}$)**.** The following conventions about the component $\mathcal{P} = (\mathcal{G}, P)$ of our chains will be observed during our development:

1. According to the usual terminology [35], we often call *pairs* the rules $u \to v \in \mathcal{P}$.
2. We have to mark terms $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ before *connecting* them to the instance $\sigma(u_{i+1})$ of the left-hand side of the next pair. Since marked symbols $f^\sharp$ are *fresh* (with respect to the signature $\mathcal{F}$ of the TRS $\mathcal{R}$), we also assume that $\mathcal{D}^\sharp \cap \mathcal{F} = \varnothing$ and $\mathcal{D}^\sharp \subseteq \mathcal{G}$.
3. We assume that $\mathcal{P}$ contains a *finite* set of rules. This is essential in many proofs.

In the following, the pairs in a CS-TRS $(\mathcal{P}, \mu)$, where $\mathcal{P} = (\mathcal{G}, P)$, are partitioned according to their role in Definition 5 as follows:

$$P_{\mathcal{X}} = \{u \to v \in P \mid v \in \mathcal{V}ar(u) - \mathcal{V}ar^\mu(u)\} \text{ and } P_{\mathcal{G}} = P - P_{\mathcal{X}}$$

Remark 5 (*Collapsing pairs*)**.** Note that all pairs in $\mathcal{P}_{\mathcal{X}} = (\mathcal{G}, P_{\mathcal{X}})$ are *collapsing*. The rules in $\mathcal{P}_{\mathcal{G}} = (\mathcal{G}, P_{\mathcal{G}})$ *can be collapsing* as well: a rewrite rule $f(x) \to x \in \mathcal{P}$ with $\mu(f) = \{1\}$ does *not* belong to $\mathcal{P}_{\mathcal{X}}$ but rather to $\mathcal{P}_{\mathcal{G}}$ because x is not a migrating variable.

Despite this fact, we refer to $\mathcal{P}_\mathcal{X}$ as the set of *collapsing* pairs in $\mathcal{P}$ because its intended role in Definition 5 is capturing the computational behavior of collapsing CSDPs in $\mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$.

Remark 6 (*Notation for chains*)**.** In general, a $(\mathcal{P}, \mathcal{R}, \mu)$-chain can be written as follows:

$$\sigma(u_1) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_2 \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

where, for all $i \geqslant 1$ and $u_i \to v_i \in \mathcal{P}$,

1. if $u_i \to v_i \notin \mathcal{P}_\mathcal{X}$, then $t_i = \sigma(v_i)$,
2. if $u_i \to v_i \in \mathcal{P}_\mathcal{X}$, then $t_i = s_i^\sharp$ for some term s_i such that $\sigma(v_i) = C_i[s_i]_{p_i}$ for some $C_i[\,]_{p_i}$ such that $p_i \in \mathcal{P}os^\mu(C_i[\,]_{p_i})$, and $\mathrm{sprefix}_{C_i[\,]_{p_i}}(p_i) \subseteq \mathcal{F}$.

This is denoted in a compact way by $\sigma(u_i) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_i$ emphasizing that there is a $\mathcal{P}$-step followed by either an equality step (as in (1)) or by μ-replacing projection steps (restricted to symbols in $\mathcal{F}$) plus a marking operation (as in (2)) depending on the considered pair $u_i \to v_i$.

5.1. Properties of some particular chains

In the following, we let $\mathcal{NHT}_\mathcal{P}(\mathcal{R}, \mu) \subseteq \mathcal{NHT}(\mathcal{R}, \mu)$ (or just $\mathcal{NHT}_\mathcal{P}$) be as follows:

$$\mathcal{NHT}_\mathcal{P}(\mathcal{R}, \mu) = \left\{ t \in \mathcal{NHT}(\mathcal{R}, \mu) \mid \exists u \to v \in \mathcal{P}, \exists \theta, \theta', \theta(t^\sharp) \hookrightarrow^*_{\mathcal{R},\mu} \theta'(u) \right\}$$

This set contains the narrowable hidden terms that 'connect' with pairs in $\mathcal{P}$.

Remark 7. Note that $\mathcal{NHT}_\mathcal{P}(\mathcal{R}, \mu)$ is not computable, in general, due to the need for checking the reachability of $\theta'(u)$ from $\theta(t^\sharp)$ using *CSR*. Suitable (over)*approximations* are discussed below (see Remark 10).

We let $\mathcal{P}^1_\mathcal{X}$ denote the subTRS of $\mathcal{P}_\mathcal{X}$ containing the rules whose migrating variables occur on non-μ-replacing immediate subterms in the left-hand side:

$$\mathcal{P}^1_\mathcal{X} = \{f(u_1, \ldots, u_k) \to x \in \mathcal{P}_\mathcal{X} \mid \exists i, 1 \leqslant i \leqslant k, i \notin \mu(f), x \in \mathcal{V}ar(u_i)\}$$

Proposition 8. *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$.*

1. *If $\mathcal{NHT}_\mathcal{P} = \varnothing$, then every infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain is an infinite minimal $(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)$-chain and there is no infinite minimal $(\mathcal{P}_\mathcal{X}, \mathcal{R}, \mu)$-chain.*
2. *If $\mathcal{P} = \mathcal{P}^1_\mathcal{X}$, then there is no infinite $(\mathcal{P}, \mathcal{R}, \mu)$-chain.*

Proof.

1. By contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain containing any $u_i \to v_i \in \mathcal{P}_\mathcal{X}$. By Definition 5, such a pair must be followed by a pair $u_{i+1} \to v_{i+1} \in \mathcal{P}$ such that $\theta_i(\bar{s}_i^\sharp) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$ for some $\bar{s}_i \in \mathcal{NHT}$ and substitution θ_i. Therefore, $t'_i \in \mathcal{NHT}_\mathcal{P}$, but $\mathcal{NHT}_\mathcal{P} = \varnothing$, leading to a contradiction.
2. By contradiction. Assume that there is an infinite chain that only uses dependency pairs $u_i \to x_i \in \mathcal{P}^1_\mathcal{X}$ for all $i \geqslant 1$. Let $f_i = root(u_i)$ for $i \geqslant 1$. Then, by definition of $\mathcal{P}^1_\mathcal{X}$, for all $i \geqslant 1$, there is $j_i \in \{1, \ldots, ar(f_i)\} - \mu(f_i)$ such that $u_i|_{j_i} \trianglerighteq x_i$. According to Definition 5, we have that $\sigma(u_i)|_{j_i} \trianglerighteq \sigma(x_i) \trianglerighteq_\mu s_i$ for some term s_i such that $s_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$. Since $root(s_i^\sharp) \in \mathcal{D}^\sharp \subseteq \mathcal{G}$ and $\mathcal{D}^\sharp \cap \mathcal{F} = \varnothing$ (Remark 4), no μ-rewriting step is possible at the root of $s_i^\sharp$. Thus, $root(s_i^\sharp) = root(u_{i+1}) = f_{i+1}$ and $j_{i+1} \notin \mu(f_{i+1})$. Since no μ-rewriting step is possible on the j_{i+1}th immediate subterm $s_i^\sharp|_{j_{i+1}}$ of $s_i^\sharp$, it follows that $s_i^\sharp|_{j_{i+1}} = \sigma(u_{i+1})|_{j_{i+1}} \trianglerighteq \sigma(x_{i+1})$, i.e., $\sigma(x_i) \rhd \sigma(x_{i+1})$ for all $i \geqslant 1$. We get an infinite sequence $\sigma(x_1) \rhd \sigma(x_2) \rhd \cdots$ which contradicts well-foundedness of $\rhd$. □

The following proposition establishes some important 'basic' cases of (absence of) infinite context-sensitive chains of pairs which are used later.

Proposition 9. *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$.*

1. *If $P = \varnothing$, then every $(\mathcal{P}, \mathcal{R}, \mu)$-chain is empty.*

2. *If* $R = \emptyset$, *then there is no infinite* $(\mathcal{P}_\mathcal{X}, \mathcal{R}, \mu)$*-chain.*
3. *Let* $u \to v \in \mathcal{P}_\mathcal{G}$ *be such that* $v = \theta(u)$. *Then, there is an infinite* $(\mathcal{P}, \mathcal{R}, \mu)$*-chain.*

Proof.

1. Obvious, by Definition 5.
2. By contradiction. If there is an infinite $(\mathcal{P}_\mathcal{X}, \mathcal{R}, \mu)$-chain, then, since there is no rule in $\mathcal{R}$, there is a substitution σ such that

$$\sigma(u_1) \hookrightarrow_{\mathcal{P},\mu} \sigma(x_1) \trianglerighteq^\sharp_\mu t_1 = \sigma(u_2) \hookrightarrow_{\mathcal{P},\mu} \sigma(x_2) \trianglerighteq^\sharp_\mu t_2 = \sigma(u_3) \cdots$$

   where $t_i = s_i^\sharp$ for some terms $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\sigma(x_i) = C_i[s_i]_{p_i}$ for some $C_i[\ ]_{p_i}$ and $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$ such that $sprefix(p_i) \subseteq \mathcal{F}$ for $i \geqslant 1$. Since $x_i \in \mathcal{V}ar(u_i)$ and u_i is not a variable, we have $u_i \rhd x_i$; hence, $\sigma(u_i) \rhd \sigma(x_i)$ (by stability of $\rhd$) and also $\sigma(u_i) \rhd s_i$ for all $i \geqslant 1$. Since s_i and $\sigma(u_{i+1})$ only differ in the root symbol, we can actually say that $s_i \rhd s_{i+1}$ for all $i \geqslant 1$. Thus, we obtain an infinite sequence $s_1 \rhd s_2 \rhd \cdots$ that contradicts the well-foundedness of $\rhd$.
3. Trivial. □

The following example shows that Proposition 9(2) does not hold for TRSs $\mathcal{P}$ with arbitrary rules.

Example 12. Consider $\mathcal{P} = \{\mathtt{F}(x) \to x, \mathtt{G}(x) \to \mathtt{F}(\mathtt{g}(x))\}$ together with a TRS $\mathcal{R}$ with an emtpy set of rules: $\mathcal{R} = (\{\mathtt{g}\}, \emptyset)$. Let μ be given by $\mu(f) = \emptyset$ for all $f \in \mathcal{F} \cup \mathcal{G}$. Note that $\mathcal{P}_\mathcal{X}$ consists of the pair $\mathtt{F}(x) \to x$ because $x \in \mathcal{V}ar(\mathtt{F}(x)) - \mathcal{V}ar^\mu(\mathtt{F}(x))$. Then, we have an infinite chain

$$\mathtt{F}(\mathtt{g}(x)) \hookrightarrow_{\mathcal{P},\mu} \mathtt{g}(x) \trianglerighteq^\sharp_\mu \mathtt{G}(x) \hookrightarrow_{\mathcal{P},\mu} \mathtt{F}(\mathtt{g}(x)) \hookrightarrow_{\mathcal{R},\mu} \cdots$$

Since $\mathcal{NHT} = \emptyset$, $\mathtt{g}(x)$ is not an instance of any term in $\mathcal{NHT}$. Thus, the chain is *not* minimal.

5.2. Chains of CSDPs vs. chains of DPs

The definition of chain of CSDPs differs from the one for DPs. First, we use $\hookrightarrow^*$ instead of $\to^*$ for connecting pairs. Also, we require μ-termination instead of termination for minimal chains. However, the most important difference concerns the treatment of collapsing pairs. In general (and in sharp contrast with the DP approach), the connection between the right-hand side of a collapsing pair (which is a variable, e.g., x) and the left-hand side u of the next pair in the chain depends on whether a marked *narrowable hidden term* (which is introduced by a *previous* μ-rewriting step) μ-rewrites into $\sigma(u)$. Dealing with collapsing pairs, hidden terms can be thought of as playing the role of *hidden* or *delayed* recursive paths. This fits the guiding idea of the DP approach as an analysis of rewriting-based recursion paths in function calls (as briefly discussed in Section 1).

6. Characterizing termination of *CSR* using chains of CSDPs

The following result establishes the soundness of the CSDP approach.

Theorem 2 (Soundness). *Let* $\mathcal{R}$ *be a TRS and* $\mu \in M_\mathcal{R}$. *Then,* $\mathcal{R}$ *is* μ*-terminating if there is no infinite minimal* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$*-chain.*

Proof. By contradiction. If $\mathcal{R}$ is not μ-terminating, then there is $t \in \mathcal{T}_{\infty,\mu}$ (Lemma 2). By Theorem 1, there are rules $l_i \to r_i \in \mathcal{R}$, substitutions σ_i, and terms $t_i \in \mathcal{M}_{\infty,\mu}$, for $i \geqslant 1$ such that

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}{}^* \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}{}^* \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}{}^* \cdots$$

where either (D1) $t_i = \sigma_i(s_i)$ for some s_i such that $r_i \trianglerighteq_\mu s_i$ or (D2) $\sigma_i(x_i) \trianglerighteq_\mu t_i$ for some $x_i \in \mathcal{V}ar^\mu(r_i) - \mathcal{V}ar^\mu(l_i)$ and $t_i = \theta_i(t_i')$ for some $t_i' \in \mathcal{NHT}$. Furthermore, since $t_{i-1} \overset{>\Lambda}{\hookrightarrow}{}^* \sigma_i(l_i)$ and $t_{i-1} \in \mathcal{M}_{\infty,\mu}$ (in particular, $t_0 = t \in \mathcal{T}_{\infty,\mu} \subseteq \mathcal{M}_{\infty,\mu}$), by Lemma 4, $\sigma_i(l_i) \in \mathcal{M}_{\infty,\mu}$ for all $i \geqslant 1$. Note that, since $t_i \in \mathcal{M}_{\infty,\mu}$, we have that $t_i^\sharp$ is μ-terminating (with respect to $\mathcal{R}$), because all μ-replacing subterms of t_i (hence of $t_i^\sharp$ as well) are μ-terminating and $root(t_i^\sharp)$ is not a defined symbol of $\mathcal{R}$.

First, note that $\mathsf{DP}(\mathcal{R}, \mu)$ is a TRS $\mathcal{P}$ over the signature $\mathcal{G} = \mathcal{F} \cup \mathcal{D}^\sharp$ and $\mu^\sharp \in M_{\mathcal{F}\cup\mathcal{G}}$ as required by Definition 5. Furthermore, $\mathcal{P}_\mathcal{G} = \mathsf{DP}_\mathcal{F}(\mathcal{R}, \mu)$ and $\mathcal{P}_\mathcal{X} = \mathsf{DP}_\mathcal{X}(\mathcal{R}, \mu)$. We can define an infinite minimal $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$-chain using CSDPs $u_i \to v_i$ for $i \geqslant 1$, where $u_i = l_i^\sharp$ and

1. $v_i = s_i^\sharp$ if (D1) holds. Since $t_i \in \mathcal{M}_{\infty,\mu}$, we have that $root(s_i) \in \mathcal{D}$ and, because $t_i = \sigma_i(s_i)$, by Corollary 2, $\text{REN}^\mu(s_i)$ is μ-narrowable. Furthermore, if we assume that s_i is a μ-replacing subterm of l_i (i.e., $l_i \rhd_\mu s_i$), then $\sigma_i(l_i) \rhd_\mu \sigma_i(s_i)$. Since $\sigma_i(s_i) = t_i \in \mathcal{M}_{\infty,\mu}$, this contradicts that $\sigma_i(l_i) \in \mathcal{M}_{\infty,\mu}$. Thus, $l_i \not\rhd_\mu s_i$. Hence, $u_i \to v_i \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$. Furthermore, $t_i^\sharp = \sigma_i(v_i)$ is μ-terminating. Finally, since $t_i = \sigma_i(s_i) \xhookrightarrow{>\Lambda}{}^* \sigma_{i+1}(l_{i+1})$ and $\mu^\sharp$ extends μ to $\mathcal{F} \cup \mathcal{D}^\sharp$ by $\mu^\sharp(f^\sharp) = \mu(f)$ for all $f \in \mathcal{D}$, we also have that $\sigma_i(v_i) = \sigma_i(s_i^\sharp) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma_{i+1}(u_{i+1})$.
2. $v_i = x_i$ if (D2) holds. Clearly, $u_i \to v_i \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$. As discussed above, $t_i^\sharp$ is μ-terminating. Since $\sigma_i(x_i) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $\sigma_i(x_i) \trianglerighteq_\mu t_i$, we have that $\sigma(v_i) = C_i[t_i]_{p_i}$ for some $C_i[\ ]_{p_i}$ and $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$ such that $\text{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$. Finally, since $t_i \xhookrightarrow{>\Lambda}{}^* \sigma_{i+1}(l_{i+1})$, again we have that $t_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma_{i+1}(u_{i+1})$. Furthermore, $t_i = \theta_i(t_i')$ for some $t_i' \in \mathcal{NHT}$ and substitution θ_i.

Regarding σ, w.l.o.g. we can assume that $\mathcal{V}ar(l_i) \cap \mathcal{V}ar(l_j) = \varnothing$ for all $i \neq j$, and therefore $\mathcal{V}ar(u_i) \cap \mathcal{V}ar(u_j) = \varnothing$ as well. Then, σ is given by $\sigma(x) = \sigma_i(x)$ whenever $x \in \mathcal{V}ar(u_i)$ for $i \geqslant 1$. From the discussion in (1) and (2), we conclude that the CSDPs $u_i \to v_i$ together with σ define an infinite minimal $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$-chain. This leads to a contradiction. □

Let $\mathsf{DP}^1_{\mathcal{X}}(\mathcal{R}, \mu) = \mathcal{P}^1_{\mathcal{X}}$ for $\mathcal{P} = \mathsf{DP}(\mathcal{R}, \mu)$. By Theorem 2 and Propositions 8 and 9, we have the following.

Corollary 3 (Basic μ-termination criteria)**.** *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$.*

1. *If $\mathsf{DP}(\mathcal{R}, \mu) = \varnothing$, then $\mathcal{R}$ is μ-terminating.*
2. *If $\mathcal{NHT}_{\mathsf{DP}(\mathcal{R},\mu)}(\mathcal{R}, \mu) = \varnothing$ and $\mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu) = \varnothing$, then $\mathcal{R}$ is μ-terminating.*
3. *If $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}^1_{\mathcal{X}}(\mathcal{R}, \mu)$, then $\mathcal{R}$ is μ-terminating.*

Example 13. Consider the following TRS $\mathcal{R}$ [44, Example 15]:

$$\begin{aligned}
\texttt{and}(\texttt{true}, x) &\to x & \texttt{add}(\texttt{0}, x) &\to x\\
\texttt{and}(\texttt{false}, y) &\to \texttt{false} & \texttt{add}(\texttt{s}(x), y) &\to \texttt{s}(\texttt{add}(x, y))\\
\texttt{if}(\texttt{true}, x, y) &\to x & \texttt{from}(x) &\to \texttt{cons}(x, \texttt{from}(\texttt{s}(x)))\\
\texttt{if}(\texttt{false}, x, y) &\to y & \texttt{first}(\texttt{0}, x) &\to \texttt{nil}\\
\texttt{first}(\texttt{s}(x), \texttt{cons}(y, z)) &\to \texttt{cons}(y, \texttt{first}(x, z))
\end{aligned}$$

together with the *canonical* replacement map $\mu(\texttt{cons}) = \mu(\texttt{s}) = \mu(\texttt{from}) = \varnothing$, $\mu(\texttt{add}) = \mu(\texttt{and}) = \mu(\texttt{if}) = \{1\}$, and $\mu(\texttt{first}) = \{1, 2\}$, which ensures completeness of *CSR* for computing head-normal forms[3] with $\mathcal{R}$ (see [44,46]). Then, $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}^1_{\mathcal{X}}(\mathcal{R}, \mu)$ is:

$$\begin{aligned}
\texttt{AND}(\texttt{true}, x) &\to x & \texttt{IF}(\texttt{true}, x, y) &\to x\\
\texttt{ADD}(\texttt{0}, x) &\to x & \texttt{IF}(\texttt{false}, x, y) &\to y
\end{aligned}$$

Note also that $\mathcal{NHT}_{\mathsf{DP}(\mathcal{R},\mu)} = \varnothing$. Thus, by either of the last two statements of Corollary 3, we conclude the μ-termination of $\mathcal{R}$.

The following example shows that Corollary 3(3) does *not* hold for chains consisting of *arbitrary* collapsing CSDPs.

Example 14. Consider the CS-TRS $(\mathcal{R}, \mu)$ in Example 3. Note that $\mathsf{DP}(\mathcal{R}, \mu) = \mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ (both $\mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ and $\mathsf{DP}^1_{\mathcal{X}}(\mathcal{R}, \mu)$ are empty!). We have the following infinite $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$-chain:

$$\texttt{F}(\underline{\texttt{a}}) \hookrightarrow_{\mathcal{R},\mu^\sharp} \underline{\texttt{F}(\texttt{c}(\texttt{f}(\texttt{a})))} \hookrightarrow_{\mathsf{DP}(\mathcal{R},\mu),\mu^\sharp} \texttt{F}(\underline{\texttt{a}}) \hookrightarrow_{\mathcal{R},\mu^\sharp} \cdots$$

Now, we prove that the previous CS-dependency pair approach is not only correct but also complete for proving termination of *CSR*.

Theorem 3 (Completeness)**.** *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. If $\mathcal{R}$ is μ-terminating, then there is no infinite $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$-chain.*

[3] A head-normal form is a term that cannot be rewritten to a redex.

Proof. By contradiction. If there is an infinite $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu^\sharp)$-chain, then there are a substitution σ and dependency pairs $u_i \to v_i \in \mathsf{DP}(\mathcal{R},\mu)$ such that

1. $\sigma(v_i) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$, if $u_i \to v_i \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$, and
2. if $u_i \to v_i = u_i \to x_i \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$, then there is $s_i \in \mathcal{T}(\mathcal{F},\mathcal{X})$ such that $\sigma(x_i) \trianglerighteq_\mu s_i$ and $s_i^\sharp \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$.

for $i \geqslant 1$. Now, consider the first dependency pair $u_1 \to v_1$ in the sequence:

1. If $u_1 \to v_1 \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$, then $v_1^\sharp$ is a μ-replacing subterm of the right-hand-side r_1 of a rule $l_1 \to r_1$ in $\mathcal{R}$. Therefore, $r_1 = C_1[v_1^\sharp]_{p_1}$ for some position $p_1 \in \mathcal{P}os^\mu(r_1)$ and context $C_1[\,]_{p_1}$, and we can perform the μ-rewriting step $t_1 = \sigma(u_1) \hookrightarrow_{\mathcal{R},\mu} \sigma(r_1) = \sigma(C_1)[\sigma(v_1^\sharp)]_{p_1} = s_1$, where $\sigma(v_1^\sharp)^\sharp = \sigma(v_1) \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_2)$ and $\sigma(u_2)$ initiates an infinite $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu^\sharp)$-chain. Note that $p_1 \in \mathcal{P}os^\mu(s_1)$.
2. If $u_1 \to x \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$, then there is a rule $l_1 \to r_1$ in $\mathcal{R}$ such that $u_1 = l_1^\sharp$, and $x \in \mathcal{V}ar^\mu(r_1) - \mathcal{V}ar^\mu(l_1)$, i.e., $r_1 = C_1[x]_{q_1}$ for some $q_1 \in \mathcal{P}os^\mu(r_1)$. Furthermore, since $\sigma(x) = C_1'[s]_{p_1'}$ for some term s, $C_1'[\,]_{p_1'}$ and $p_1' \in \mathcal{P}os^\mu(C_1'[\,]_{p_1'})$ such that $s^\sharp \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_2)$, we can perform the μ-rewriting step $t_1 = \sigma(l_1) \hookrightarrow_{\mathcal{R},\mu} \sigma(r_1) = \sigma(C_1)[C_1'[s]_{p_1'}]_{q_1} = s_1$ where $s^\sharp \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_2)$ (hence $s \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathcal{R},\mu} u_2^\sharp$) and $\sigma(u_2)$ initiates an infinite $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu^\sharp)$-chain. Note that $p_1 = q_1 \cdot p_1' \in \mathcal{P}os^\mu(s_1)$ (use Proposition 1).

Since $\mu^\sharp(f^\sharp) = \mu(f)$, and $p_1 \in \mathcal{P}os^\mu(s_1)$, we have that $s_1 \hookrightarrow^*_{\mathcal{R},\mu} t_2[\sigma(u_2)]_{p_1} = t_2$ and $p_1 \in \mathcal{P}os^\mu(t_2)$. Thus, we can build an infinite μ-rewrite sequence $t_1 \hookrightarrow_{\mathcal{R},\mu} s_1 \hookrightarrow^*_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \cdots$ which contradicts the μ-termination of $\mathcal{R}$. □

Proposition 9(3) suggests a simple checking of *non-μ-termination*.

Corollary 4 (Non-μ-termination criterion). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. If there is $u \to v \in \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu)$ such that $v' = \theta(u)$ for some substitution θ and renamed version v' of v, then $\mathcal{R}$ is not μ-terminating.*

As a corollary of Theorems 2 and 3, we have:

Corollary 5 (Characterization of μ-termination). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Then, $\mathcal{R}$ is μ-terminating if and only if there is no infinite minimal $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu^\sharp)$-chain.*

7. Mechanizing proofs of μ-termination using CSDPs

Over the last 10 years, the dependency pair method has evolved to a powerful technique for proving termination of TRSs in practice. In the DP-approach [10], the starting point is a TRS $\mathcal{R}$ from which a set of dependency pairs $\mathsf{DP}(\mathcal{R})$ is obtained. Then, these dependency pairs are organized in a dependency graph $\mathsf{DG}(\mathcal{R})$ whose nodes are the pairs in $\mathsf{DP}(\mathcal{R})$ and where the arcs are obtained by investigating possible *rewriting connections* between (instances of) the right-hand sides of the pairs and (instances of) the left-hand sides of other (not necessarily distinct) pairs. The cycles of the graph are analyzed to show that no infinite chains of DPs can be obtained from them [25]. In this sense, the treatment of strongly connected components of the graph (SCCs) instead of cycles [38,39] brought an important improvement to the practical use of this approach.

In the DP-approach, the components $u_i \to v_i$ of the chains (or cycles) are dependency pairs, i.e., $u_i \to v_i \in \mathsf{DP}(\mathcal{R})$ for all $i \geqslant 1$. Since they only make sense when an underlying TRS $\mathcal{R}$ is given as the source of the dependency pairs, transforming DPs is possible (the *narrowing* transformation is already described in [10]) but only as a final step because, afterwards, they are no longer dependency pairs of the original TRS. The dependency pair framework [33,35] solves this problem in a clear way, leading to a more powerful mechanization of termination proofs. The central notion now is that of *DP problem* [35, p. 158]: given a TRS $\mathcal{R}$ and a set of pairs $\mathcal{P}$, the goal is to verify the absence of infinite (minimal) chains. In this case, the DP problem is called *finite*. Termination of a TRS $\mathcal{R}$ is addressed as a DP problem[4] $(\mathcal{P},\mathcal{R})$ where $\mathcal{P} = \mathsf{DP}(\mathcal{R})$: $\mathcal{R}$ is terminating if this problem is finite. The most important notion regarding the mechanization of the proofs is the notion of *processor*. Formally, *a DP processor is a function* Proc *that takes a DP problem as input and returns a new set of DP problems that then have to be solved instead. Alternatively, it can also return* "no" [35, p. 159]. In the following, we adapt the notions of [35] to *CSR*.

[4] The original definition in [35] includes an extra parameter e, which specifies two kinds of problems: $e = \mathbf{t}$ for termination problems, and $e = \mathbf{i}$ for *innermost* termination problems.

7.1. CS problems, CS processors, and the CSDP-framework

In our definition of *DP problem* for *CSR*, we prefer to avoid '*DP*' because, as discussed above, dependency pairs (as such) are relevant in the theoretical framework only for investigating a particular problem (termination of TRSs), whereas some transformations can yield sets of pairs which are no longer dependency pairs of the underlying TRS.

Definition 6 (*CS problem*)**.** A CS problem τ is a tuple $\tau = (\mathcal{P}, \mathcal{R}, \mu)$, where $\mathcal{R}$ and $\mathcal{P}$ are TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. The CS problem τ is finite if there is no infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain. The CS problem τ is infinite if $\mathcal{R}$ is non-μ-terminating or there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain.

Remark 8. As in the standard DP framework (see the discussion and further motivation in [35, p. 159]), the inclusion of the case when $\mathcal{R}$ is nonterminating as part of the definition of infinite problem is essential for dealing with some specific transformations of CS problems (see Theorems 8 and 16).

Definition 7 (*CS processor*)**.** A CS processor Proc is a mapping from CS problems into sets of CS problems. Alternatively, it can also return "no". A CS processor Proc is

- *sound* if for all CS problems τ, we have that (1) τ is finite whenever $\mathsf{Proc}(\tau) \neq \mathsf{no}$ and (2) $\forall \tau' \in \mathsf{Proc}(\tau)$, τ' is finite.
- *complete* if for all CS problems τ, we have that (1) τ is infinite whenever $\mathsf{Proc}(\tau) = \mathsf{no}$ or (2) $\exists \tau' \in \mathsf{Proc}(\tau)$ such that τ' is infinite.

A (sound) processor transforms DP problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. Here, 'simpler' usually means that fewer pairs are involved. Soundness is essential for proving *termination*. Completeness is necessary for proving *nontermination*.

Processors are used in a *divide and conquer* scheme to incrementally simplify the original CS problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. The trivial case comes when the set of pairs $\mathcal{P}$ becomes empty. Then, no infinite chain is possible, and we can provide a *positive* answer yes to the CS problem which is propagated upwards to the original problem in the root of the decision tree. In some cases, it is also possible to witness the existence of infinite chains for a given CS problem; then a *negative* answer no can be provided and propagated upwards.

Theorem 4 (CSDP-framework)**.** *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We construct a tree whose nodes are labeled with CS problems or* "yes" *or* "no"*, and whose root is labeled with* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\sharp})$*. For every inner node labeled with τ, there is a sound processor* Proc *satisfying one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child that is labeled with* "no"*.*
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child that is labeled with* "yes"*.*
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$*, and the children of the node are labeled with the CS problems in* $\mathsf{Proc}(\tau)$*.*

If all leaves of the tree are labeled with "yes"*, then $\mathcal{R}$ is μ-terminating. Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is not μ-terminating.*

Propositions 8 and 9 are the basis for the following sound and complete processors, which provide some *base* cases for our proofs of termination of *CSR*.

Theorem 5 (Basic CS processors)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Then, the processors* Proc_{Fin} *and* Proc_{Inf} *given by*[5]

$$\mathsf{Proc}_{Fin}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \varnothing & \text{if } P = \varnothing \vee \mathcal{P} = \mathcal{P}^1_{\mathcal{X}} \vee (R = \varnothing \wedge \mathcal{P} = \mathcal{P}_{\mathcal{X}}); \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

$$\mathsf{Proc}_{Inf}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \mathsf{no} & \text{if } v = \theta(u) \\ & \text{for some } u \to v \in \mathcal{P}_{\mathcal{G}} \text{ and substitution } \theta; \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

are sound and complete.

[5] In the following, we often write $\mathsf{Proc}(\mathcal{P}, \mathcal{R}, \mu)$ instead of $\mathsf{Proc}((\mathcal{P}, \mathcal{R}, \mu))$ to avoid duplicated parentheses.

In the following sections, we describe several sound and (most of them) complete CS processors.

8. Context-sensitive dependency graph

In the dependency pairs approach [10], a *dependency graph* $\mathsf{DG}(\mathcal{R})$ is associated to the TRS $\mathcal{R}$. The nodes of $\mathsf{DG}(\mathcal{R})$ are the dependency pairs in $\mathsf{DP}(\mathcal{R})$; there is an arc from a dependency pair $u \to v$ to a dependency pair $u' \to v'$ such that $\mathcal{V}ar(u) \cap \mathcal{V}ar(u') = \varnothing$ if $\theta(v) \to_{\mathcal{R}}^* \theta(u')$ for some substitution θ. In [35], a more general notion of *graph of pairs* $\mathsf{DG}(\mathcal{P}, \mathcal{R})$ associated to a set of pairs $\mathcal{P}$ and a TRS $\mathcal{R}$ is considered. Pairs in $\mathcal{P}$ are now used as the nodes of the graph, but they are connected by $\mathcal{R}$-rewriting in the same way [35, Definition 7]. The analysis of the cycles in the graph that is built from such pairs is useful for investigating the existence of infinite (minimal) chains of pairs. In the following section, we take into account these points to provide an appropriate definition of context-sensitive (dependency) graph.

8.1. Definition of the context-sensitive dependency graph

Given TRSs $\mathcal{R}$ and $\mathcal{P}$ and a replacement map $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$, we want to obtain a notion of graph that is able to represent all infinite *minimal* chains of pairs as given in Definition 5.

Definition 8 (*Context-sensitive graph of pairs*)**.** Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. The *context-sensitive (CS-)graph* $\mathsf{G}(\mathcal{P}, \mathcal{R}, \mu)$ has $\mathcal{P}$ as the set of nodes. Given $u \to v, u' \to v' \in \mathcal{P}$, there is an arc from $u \to v$ to $u' \to v'$ if $u \to v, u' \to v'$ is a minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain for some substitution σ.

In termination proofs, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [39]. A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle that is not contained in any other cycle. The following result justifies the use of SCCs for proving the absence of infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chains.

Theorem 6 (SCC processor)**.** *Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. Then, the processor Proc_{SCC} given by*

$$\mathsf{Proc}_{SCC}(\mathcal{P}, \mathcal{R}, \mu) = \{(\mathcal{Q}, \mathcal{R}, \mu) \mid \mathcal{Q} \text{ are the pairs of an SCC in } \mathsf{G}(\mathcal{P}, \mathcal{R}, \mu)\}$$

is sound and complete.

Proof. We prove soundness by contradiction. Assume that Proc_{SCC} is not sound. Then, there is a CS problem $\tau = (\mathcal{P}, \mathcal{R}, \mu)$ such that, for all $\tau' \in \mathsf{Proc}_{SCC}(\tau)$, τ' is finite but τ is not finite. Thus, there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A. Since $\mathcal{P}$ contains a finite number of pairs, there is $\mathcal{P}' \subseteq \mathcal{P}$ and a tail B of A, which is an infinite minimal $(\mathcal{P}', \mathcal{R}, \mu)$-chain where all pairs in $\mathcal{P}'$ are infinitely often used. According to Definition 8, this means that $\mathcal{P}'$ is a cycle in $\mathsf{G}(\mathcal{P}, \mathcal{R}, \mu)$. Hence $\mathcal{P}'$ belongs to some SCC with nodes in $\mathcal{Q}$, i.e., $\mathcal{P}' \subseteq \mathcal{Q}$. Thus, B is an infinite minimal $(\mathcal{Q}, \mathcal{R}, \mu)$-chain, i.e., $\tau' = (\mathcal{Q}, \mathcal{R}, \mu)$ is not finite. Since $\tau' \in \mathsf{Proc}_{SCC}(\tau)$, we obtain a contradiction.

With regard to completeness, since $\mathcal{Q} \subseteq \mathcal{P}$ for some SCC in $\mathsf{G}(\mathcal{P}, \mathcal{R}, \mu)$ with nodes in $\mathcal{Q}$, every infinite minimal $(\mathcal{Q}, \mathcal{R}, \mu)$-chain is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain. Hence, the processor is complete as well. □

As a consequence of this theorem, we can *separately* work with the strongly connected components of $\mathsf{G}(\mathcal{P}, \mathcal{R}, \mu)$, disregarding other parts of the graph. Now we can use these notions to introduce the context-sensitive dependency graph.

Definition 9 (*Context-sensitive dependency graph*)**.** Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. The *Context-Sensitive Dependency Graph* (CSDG) for $\mathcal{R}$ and μ is $\mathsf{DG}(\mathcal{R}, \mu) = \mathsf{G}(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$.

8.2. Estimating the CS-dependency graph

In general, the context-sensitive graph is *not* computable: it involves reachability of $\sigma(u')$ from $\sigma(v)$ (for $u \to v \in \mathcal{P}_{\mathcal{G}}$) or $\sigma(t^\sharp)$ (for $t \in \mathcal{NHT}_{\mathcal{P}}$) using *CSR*. Since the reachability problem for *CSR* is undecidable, we need to use some approximation of it.

Remark 9. Several estimations of the dependency graph were investigated in [10,34,39,55,56]. The first one, introduced in [10], was adapted to *CSR* in [3].

Following [34], we describe how to approximate the CS-dependency graph of a CS-TRS. Given a TRS $\mathcal{R}$ and a replacement map μ, we let $\mathrm{TCAP}_{\mathcal{R}}^{\mu}$ be as follows:

$$\begin{aligned}\text{TCAP}^{\mu}_{\mathcal{R}}(x) &= y \text{ if } x \text{ is a variable, and}\\ \text{TCAP}^{\mu}_{\mathcal{R}}(f(t_1,\ldots,t_k)) &= \begin{cases} f([t_1]^f_1,\ldots,[t_k]^f_k) & \text{if } f([t_1]^f_1,\ldots,[t_k]^f_k) \text{ does not unify}\\ & \text{with } l \text{ for any } l \to r \text{ in } \mathcal{R}\\ y & \text{otherwise}\end{cases}\end{aligned}$$

where y is intended to be a new, fresh variable that has not yet been used and given a term s, $[s]^f_i = \text{TCAP}^{\mu}_{\mathcal{R}}(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that l shares no variable with $f([t_1]^f_1,\ldots,[t_k]^f_k)$ when the unification is attempted. Function $\text{TCAP}^{\mu}_{\mathcal{R}}$ is intended to provide a suitable approximation of the aforementioned $(\mathcal{R},\mu)$-reachability problems by means of *unification*. The following result formalizes the correctness of this approach.

Proposition 10. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. Let $t, u \in \mathcal{T}(\mathcal{F},\mathcal{X})$ be such that $\mathcal{V}ar(t) \cap \mathcal{V}ar(u) = \varnothing$. If $\theta(t) \hookrightarrow^* \theta(u)$ for some substitution θ, then $\text{TCAP}^{\mu}_{\mathcal{R}}(t)$ and u unify.*

Proof. In the following, we let $s = \text{TCAP}^{\mu}_{\mathcal{R}}(t)$. Note that, since $\mathcal{V}ar(t) \cap \mathcal{V}ar(u) = \varnothing$, we also have $\mathcal{V}ar(s) \cap \mathcal{V}ar(u) = \varnothing$. Clearly, $t = \sigma(s)$ for some substitution σ. We proceed by induction on the length m of the sequence from $\theta(t)$ to $\theta(u)$.

1. If $m = 0$, then $\theta(t) = \theta(\sigma(s)) = \theta(u)$. Since $\mathcal{V}ar(s) \cap \mathcal{V}ar(u) = \varnothing$, we can write $\theta(u) = \theta(\sigma(u))$, i.e., s and u unify.
2. If $m > 0$, then we have $\theta(t) \hookrightarrow t' \hookrightarrow^* \theta(u)$. Let $p \in \mathcal{P}os^{\mu}(\theta(t))$ be the position where the μ-rewrite step $\theta(t) \hookrightarrow t'$ is performed. By definition of $\text{TCAP}^{\mu}_{\mathcal{R}}$, $s = s[z]_q$ for some fresh variable z and position q such that $q \leqslant p$. We can write $\theta(t) = \theta(s)$. Furthermore, since z is a fresh variable, we can write $t' = \theta(s)$ if we assume that $\theta(z) = t'|_q$. Thus, $\theta(s) \hookrightarrow^* \theta(u)$ in $m-1$ steps. By the induction hypothesis, $\text{TCAP}^{\mu}_{\mathcal{R}}(s)$ and u unify. Since $\text{TCAP}^{\mu}_{\mathcal{R}}(s) = \text{TCAP}^{\mu}_{\mathcal{R}}(\text{TCAP}^{\mu}_{\mathcal{R}}(t))$ and $\text{TCAP}^{\mu}_{\mathcal{R}}(\text{TCAP}^{\mu}_{\mathcal{R}}(t))$ is just a renaming of $\text{TCAP}^{\mu}_{\mathcal{R}}(t)$, the conclusion follows. □

According to Proposition 10, given terms $t, u \in \mathcal{T}(\mathcal{F},\mathcal{X})$ that share no variable, and a substitution θ, the reachability of $\theta(u)$ from $\theta(t)$ by μ-rewriting can be *approximated* as unification of $\text{TCAP}^{\mu}_{\mathcal{R}}(t)$ and u. Thus, taking into account Definitions 5 and 8, we have the following.

Definition 10 (*Estimated context-sensitive graph of pairs*)**.** Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. The *estimated* CS-graph associated to $\mathcal{R}$ and $\mathcal{P}$ (denoted $\mathsf{EG}(\mathcal{P},\mathcal{R},\mu)$) has $\mathcal{P}$ as the set of nodes and the arcs that connect them as follows:

1. There is an arc from $u \to v \in \mathcal{P}_{\mathcal{G}}$ to $u' \to v' \in \mathcal{P}$ if $\text{TCAP}^{\mu}_{\mathcal{R}}(v)$ and u' unify.
2. There is an arc from $u \to v \in \mathcal{P}_{\mathcal{X}}$ to $u' \to v' \in \mathcal{P}$ if there is $t \in \mathcal{NHT}(\mathcal{R},\mu)$ such that $\text{TCAP}^{\mu}_{\mathcal{R}}(t^{\sharp})$ and u' unify.

As a consequence of Proposition 10, we have the following.

Corollary 6 (Approximation of the context-sensitive graph)**.** *Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. The estimated CS-graph $\mathsf{EG}(\mathcal{P},\mathcal{R},\mu)$ contains the CS-graph $\mathsf{G}(\mathcal{P},\mathcal{R},\mu)$.*

Therefore, we have the following *estimated* CSDG: $\mathsf{EDG}(\mathcal{R},\mu) = \mathsf{EG}(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mu^{\sharp})$.

Remark 10. Proposition 10 also provides *estimations* for $\mathcal{NHT}_{\mathcal{P}}$: if $t \in \mathcal{NHT}_{\mathcal{P}}$, then $\text{TCAP}^{\mu}_{\mathcal{R}}(t^{\sharp})$ and u unify for some $u \to v \in \mathcal{P}$. In the following, we compute $\mathcal{NHT}_{\mathcal{P}}$ in this way.

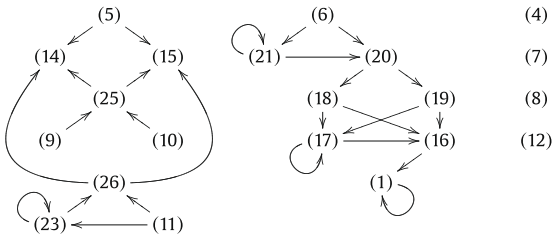


Fig. 5. Context-sensitive dependency graph for the CS-TRS in Example 1.

Example 15. Consider again the CS-TRS $(\mathcal{R}, \mu)$ in Example 1. Note that

$$\mathcal{NHT}_{\mathsf{DP}(\mathcal{R},\mu)}(\mathcal{R}, \mu^\sharp) = \{\mathtt{oddNs}, \mathtt{incr}(\mathtt{oddNs}), \mathtt{incr}(x), \mathtt{zip}(xs, ys), \mathtt{rep2}(xs)\}$$

The (estimated) CSDG in Fig. 5 has *four* cycles, each of which contains a single pair. We transform the CS problem $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ into a set

$$\mathsf{Proc}_{SCC}(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp) = \left\{ (\{(1)\}, \mathcal{R}, \mu^\sharp),\ (\{(17)\}, \mathcal{R}, \mu^\sharp),\ (\{(21)\}, \mathcal{R}, \mu^\sharp),\ (\{(23)\}, \mathcal{R}, \mu^\sharp) \right\}$$

which contains four new (but very simple) CS problems.

Remark 11 (*CSDG vs. DG*)**.** Consider again $\mathcal{R}$ and μ as in Example 1. Pairs (9) and (10) belong to both $\mathsf{DG}(\mathcal{R})$ (see Fig. 3) and $\mathsf{DG}(\mathcal{R}, \mu)$ (see Fig. 5). However, they are *not* equally connected in $\mathsf{DG}(\mathcal{R})$ and $\mathsf{DG}(\mathcal{R}, \mu)$. The reason is that the collapsing pair (25), that is *not* a node of $\mathsf{DG}(\mathcal{R})$, originates an *incoming* arc from both (9) and (10).

9. Treating collapsing pairs

The following result shows how to *safely* transform collapsing pairs into noncollapsing ones in some particular cases.

Theorem 7 (Removing collapsing pairs)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let $\mathcal{P}' = (\mathcal{G}', P')$ where $P' = (P - P_{\mathcal{X}}) \cup Q$ for $Q = \{u \to t^\sharp \mid u \to x \in \mathcal{P}_{\mathcal{X}}, t \in \mathcal{NHT}_{\mathcal{P}}\}$, $\mathcal{G}' = \mathcal{G}$ if $Q = \varnothing$, and $\mathcal{G}' = \mathcal{F} \cup \mathcal{G}$ if $Q \neq \varnothing$. Then, the processor Proc_{gNHT} given by*

$$\mathsf{Proc}_{gNHT}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}', \mathcal{R}, \mu)\} & \text{if } \mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu) \subseteq \mathcal{T}(\mathcal{F}) \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound.

Proof. First, note that $\mathcal{P}'$ is a TRS: the new rules in Q are of the form $u \to t^\sharp$ for $t \in \mathcal{NHT}_{\mathcal{P}}$. Since $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$, we trivially have $\mathcal{V}ar(t^\sharp) \subseteq \mathcal{V}ar(u)$, i.e., $u \to t^\sharp$ is a rewrite rule. Furthermore, whenever $Q \neq \varnothing$, $\mathcal{G}'$ is the union of $\mathcal{F}$ and $\mathcal{G}$ to reflect the use of symbols in $\mathcal{F}$ coming from terms $t^\sharp$ for $t \in \mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu)$). Since we assume that $\mathcal{D}^\sharp \subseteq \mathcal{G}$ (Remark 4), $root(t^\sharp) \in \mathcal{D}^\sharp \subseteq \mathcal{G} \subseteq \mathcal{G}'$.

We prove that the existence of an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain implies the existence of an infinite minimal $(\mathcal{P}', \mathcal{R}, \mu)$-chain. Consider an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain:

$$\sigma(u_1) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_2 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_3) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu \cdots$$

for some substitution σ, where, according to Definition 5, for all $i \geqslant 1$, t_i is μ-terminating and, (1) if $u_i \to v_i \in \mathcal{P}_{\mathcal{G}}$, then $t_i = \sigma(v_i)$ and (2) if $u_i \to v_i = u_i \to x_i \in \mathcal{P}_{\mathcal{X}}$, then $t_i = s_i^\sharp$ for some s_i such that $\sigma(x_i) \trianglerighteq_\mu s_i$ and $s_i = \theta_i(\bar{s}_i)$ for some $\bar{s}_i \in \mathcal{NHT}$ and substitution θ_i. Actually, since $t_i = s_i^\sharp = \theta_i(\bar{s}_i)^\sharp = \theta_i(\bar{s}_i^\sharp)$ and $t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$, we can further say that $\bar{s}_i \in \mathcal{NHT}_{\mathcal{P}}$.

In case (2), since $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$, we have $t_i = s_i^\sharp = \theta_i(\bar{s}_i^\sharp) = \bar{s}_i^\sharp$, i.e., $t_i \in \mathcal{NHT}_{\mathcal{P}}$. Thus, we can use $u_i \to t_i \in \mathcal{Q}$ instead of $u_i \to x_i \in \mathcal{P}_{\mathcal{X}}$, because we still have $t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$. In this way, by replacing each $u_i \to x_i \in \mathcal{P}_{\mathcal{X}}$ by the corresponding $u_i \to t_i \in \mathcal{Q}$, each step $\sigma(u_i) \hookrightarrow_{\mathcal{P},\mu} \circ \trianglerighteq^\sharp_\mu t_i$ becomes a step $\sigma(u_i) \hookrightarrow_{\mathcal{P}',\mu} t_i$, whereas steps $\sigma(u_i) \hookrightarrow_{\mathcal{P},\mu} \sigma(v_i) = t_i$ for $u_i \to v_i \in \mathcal{P}_{\mathcal{G}}$ remain unchanged. Thus, we obtain an infinite minimal $(\mathcal{P}', \mathcal{R}, \mu)$-chain, as desired. □

Note that no pair in $\mathcal{P}'$ in Theorem 7 is collapsing. Unfortunately, Proc_{gNHT} is *not* complete.

Example 16. Consider the following TRS:

$$\begin{aligned} \mathtt{b} &\to \mathtt{f}(\mathtt{c}(\mathtt{b})) \\ \mathtt{f}(x) &\to x \end{aligned}$$

together with the replacement map μ given by $\mu(\mathtt{f}) = \mu(\mathtt{c}) = \varnothing$. $\mathsf{DP}(\mathcal{R}, \mu)$ is:

$$\begin{aligned} \mathtt{B} &\to \mathtt{F}(\mathtt{c}(\mathtt{b})) \\ \mathtt{F}(x) &\to x \end{aligned}$$

and $\mathcal{NHT}_{\mathsf{DP}(\mathcal{R},\mu)} = \{\mathtt{b}\}$. There is no infinite $(\mathcal{P}, \mathcal{R}, \mu^\sharp)$-chain for $\mathcal{P} = \mathsf{DP}(\mathcal{R}, \mu)$, i.e., $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ is finite and $\mathcal{R}$ μ-terminating. However, with $\mathcal{P}'$ as in Theorem 7:

$$\begin{aligned}\mathtt{B} &\to \mathtt{F}(\mathtt{c}(\mathtt{b}))\\ \mathtt{F}(x) &\to \mathtt{B}\end{aligned}$$

we have an infinite minimal $(\mathcal{P}', \mathcal{R}, \mu^\sharp)$-chain, i.e, $(\mathcal{P}', \mathcal{R}, \mu^\sharp)$ is *not* finite.

The following processor provides a sound and complete *transformation* of collapsing pairs into noncollapsing pairs.

Theorem 8 (Transforming collapsing pairs). *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let $u \to x \in \mathcal{P}_\mathcal{X}$ and*

$$\begin{aligned}P_u = {} & \{u \to U(x)\}\\ & \cup \{U(f(x_1, \ldots, x_k)) \to U(x_i) \mid f \in \mathcal{F}, i \in \mu(f)\}\\ & \cup \{U(t) \to t^\sharp \mid t \in \mathcal{NHT}_\mathcal{P}\}\end{aligned}$$

where U is a fresh symbol. Let $\mathcal{P}' = (\mathcal{G} \cup \{U\}, P')$ where $P' = (P - \{u \to x\}) \cup P_u$, and μ' which extends μ by $\mu'(U) = \varnothing$. The processor Proc_{eColl} given by

$$\mathsf{Proc}_{eColl}(\mathcal{P}, \mathcal{R}, \mu) = \{(\mathcal{P}', \mathcal{R}, \mu')\}$$

is sound and complete.

Proof. With regard to soundness, we proceed by contradiction. If Proc_{eColl} is not sound, then there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain but there is no infinite minimal $(\mathcal{P}', \mathcal{R}, \mu')$-chain A. Since $\mathcal{P}$ is finite, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B

$$\sigma(u_1) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_2 \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

for some substitution σ and pairs $u_i \to v_i \in \mathcal{Q}$, and, for all $i \geqslant 1$,

1. if $v_i \notin \mathcal{X}$, then $t_i = \sigma(v_i)$,
2. if $v_i = x_i \in \mathcal{X}$, then $x_i \notin \mathcal{V}ar^\mu(u_i)$, and $\sigma(x_i) = C_i[s_i]_{p_i}$ for some context $C_i[\ ]_{p_i}$, such that $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$, $\mathsf{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$, $s_i = \theta_i(\bar{s}_i)$ for some $\bar{s}_i \in \mathcal{NHT}_\mathcal{P}$ and substitution θ_i, and $t_i = s_i^\sharp$.

For 'steps' $\sigma(u_i) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_i$ such that $u_i \to v_i \neq u \to x$, we have $u_i \to v_i \in \mathcal{P}'$. By minimality of B, t_i is $(\mathcal{R}, \mu)$-terminating. Since $t_i \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$ and $\mu'(f) = \mu(f)$ for all $f \in \mathcal{F} \cup \mathcal{G}$, t_i is $(\mathcal{R}, \mu')$-terminating, too. On the other hand, if $u_i \to v_i = u \to x$, then, since $C_i[\ ]_{p_i} \subseteq \mathcal{F}$, $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$, and by definition of P_u, we get

$$\sigma(u_i) \hookrightarrow_{P_u,\mu'} U(\sigma(v_i)) = U(C_i[s_i]_{p_i}) \hookrightarrow^*_{P_u,\mu'} U(s_i) = U(\theta_i(\bar{s}_i)) = \theta_i(U(\bar{s}_i)) \hookrightarrow_{P_u,\mu'} \theta_i(\bar{s}_i^\sharp) = s_i^\sharp = t_i$$

where all terms of the form $U(s)$ in the sequence above are $(\mathcal{R}, \mu')$-terminating: since $\mu'(U) = \varnothing$ and U does not belong to $\mathcal{F}$, $U(s)$ is in $(\mathcal{R}, \mu')$-normal form. Furthermore, by minimality of B, t_i is $(\mathcal{R}, \mu)$-terminating and, since $\mu'(f) = \mu(f)$ for all $f \in \mathcal{F} \cup \mathcal{G}$, t_i is $(\mathcal{R}, \mu')$-terminating. Therefore, we obtain an infinite minimal $(\mathcal{P}', \mathcal{R}, \mu')$-chain, leading to a contradiction.

For completeness, we consider two cases: if $\mathcal{R}$ is not μ-terminating, then all termination problems are infinite (both before and after the application of Proc_{eColl}) and there is no problem. Therefore, assume that $\mathcal{R}$ is μ-terminating and that $(\mathcal{P}, \mathcal{R}, \mu)$ is finite but there is an infinite $(\mathcal{P}', \mathcal{R}, \mu')$-chain. Again, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}'$ such that A has a tail B

$$\sigma(u_1) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu'} \circ \trianglerighteq^\sharp_{\mu'} t_1 \hookrightarrow^*_{\mathcal{R},\mu'} \sigma(u_2) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu'} \circ \trianglerighteq^\sharp_{\mu'} t_2 \hookrightarrow^*_{\mathcal{R},\mu'} \cdots$$

for some substitution σ and pairs $u_i \to v_i \in \mathcal{Q}$ where $t_i = \sigma(v_i)$ is $(\mathcal{R}, \mu')$-terminating for $i \geqslant 1$. Without loss of generality, we can assume that $\sigma(x) \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$ for all $x \in \mathcal{X}$, i.e., σ does *not* introduce any symbol U. It is not difficult to see that, for each $(\mathcal{P}', \mathcal{R}, \mu')$-chain which is based on a substitution σ' whose bindings $\sigma'(x)$ contain symbols U, there is a $(\mathcal{P}', \mathcal{R}, \mu')$-chain which uses the same pairs in $\mathcal{P}'$ and rules in $\mathcal{R}$ for the rewriting steps, but which is based on a substitution σ where the U's have been just removed from all bindings $\sigma'(x)$ to obtain $\sigma(x)$ instead.

If $u_i \to v_i \in P_u$, then, without loss of generality, we can assume that $u_i = u$ and $v_i = U(x)$. Since $\mu(U) = \varnothing$, there is $n \geqslant 0$ such that

$$\begin{aligned}
\sigma(u_i) &\overset{\Lambda}{\hookrightarrow}_{\mathcal{P}',\mu'} \sigma(U(x)) = U(\sigma(x)) = \sigma(u_{i+1}) \\
&\overset{\Lambda}{\hookrightarrow}_{\mathcal{P}',\mu'} \sigma(v_{i+1}) = \sigma(u_{i+2}) \\
&\overset{\Lambda}{\hookrightarrow}_{\mathcal{P}',\mu'} \\
&\vdots \\
&\overset{\Lambda}{\hookrightarrow}_{\mathcal{P}',\mu'} \sigma(v_{i+n}) = \sigma(U(s_{i+n+1})) \\
&\overset{\Lambda}{\hookrightarrow}_{\mathcal{P}',\mu'} \sigma(s^{\sharp}_{i+n+1}) = \sigma(t_i) \\
&\hookrightarrow^{*}_{\mathcal{R},\mu'} \sigma(u_{i+n+2})
\end{aligned}$$

where, for all $j, i+1 \leqslant j \leqslant i+n, u_j = U(f_j(x_1, \ldots, x_{i_j}, \ldots, x_{k_j})), v_j = U(x_{i_j}), i_j \in \mu(f_j)$, and $s_{i+n+1} \in \mathcal{NHT}_{\mathcal{P}}$ (by definition of Proc_{eColl}). Therefore, from the n rewriting steps that remove the $f_j \in \mathcal{F}$ for $1 \leqslant j \leqslant n$, we know that $\sigma(x) = C_i[t_{i+n+1}]_{p_i}$ with $p_i \in \mathcal{P}os^{\mu}(C_i[\ ]_{p_i})$ and $\mathsf{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$. Thus, according to Definition 5, we have: $\sigma(u_i) \overset{\Lambda}{\hookrightarrow}_{\mathcal{P},\mu} \circ \trianglerighteq^{\sharp}_{\mu} t_i$ and $t_i \hookrightarrow^{*}_{\mathcal{R},\mu} \sigma(u_{i+n+2})$. Furthermore, t_i is μ-terminating (because $\mathcal{R}$ is μ-terminating). On the other hand, if $u_i \to v_i \in \mathcal{P} - \{u \to x\}$, then we have $\sigma(u_i) \overset{\Lambda}{\hookrightarrow}_{\mathcal{P},\mu} \circ \trianglerighteq^{\sharp}_{\mu} t_i$ satisfying the conditions in Definition 5. We obtain an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain, leading again to a contradiction. □

Example 17. The use of Proc_{eColl} with $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\sharp})$ in Example 16 yields $(\mathcal{P}', \mathcal{R}, \mu')$ where $\mathcal{P}'$ consists of the following pairs:

$$\begin{array}{lll}
\mathtt{B} & \to \mathtt{F(c(b))} & \mathtt{F}(x) \to \mathtt{U}(x) \\
\mathtt{U(b)} \to \mathtt{B} &&
\end{array}$$

It is not difficult to see now that there is no infinite minimal $(\mathcal{P}', \mathcal{R}, \mu')$-chain.

10. Use of μ-reduction pairs

A reduction pair $(\gtrsim, \sqsupset)$ consists of a stable and monotonic quasi-ordering $\gtrsim$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$ [42]. The absence of infinite chains of pairs can be ensured by finding a *reduction pair* $(\gtrsim, \sqsupset)$ that is compatible with the rules and the pairs: $l \gtrsim r$ for all rewrite rules $l \to r$ and $u \gtrsim v$ or $u \sqsupset v$ for all dependency pairs $u \to v$ [10]. In the dependency pair framework, they are used to obtain *smaller* sets of pairs $\mathcal{P}' \subseteq \mathcal{P}$ by removing the *strict* pairs, i.e., those pairs $u \to v \in \mathcal{P}$ such that $u \sqsupset v$.

Stability is required for both $\gtrsim$ and $\sqsupset$ because, although we only check the left- and right-hand sides of the rewrite rules $l \to r$ (with $\gtrsim$) and pairs $u \to v$ (with $\gtrsim$ or $\sqsupset$), the chains of pairs involve *instances* $\sigma(l), \sigma(r), \sigma(u)$, and $\sigma(v)$ of rules and pairs, and we aim to conclude $\sigma(l) \gtrsim \sigma(r)$ and also $\sigma(u) \gtrsim \sigma(v)$ or $\sigma(u) \sqsupset \sigma(v)$. Monotonicity is required for $\gtrsim$ to deal with the application of rules $l \to r$ to an arbitrary depth in terms. Since the pairs are 'applied' only at the root level, no monotonicity is required for $\sqsupset$ (but, for this reason, we cannot compare the rules in $\mathcal{R}$ using $\sqsupset$). Endrullis et al. noticed that *transitivity* is not necessary for the strict component $\sqsupset$ because it is somehow 'simulated' by the compatibility requirement above [20].

In our setting, since we are interested in μ-rewriting steps only, we can relax the monotonicity requirements as follows.

Definition 11 (μ-*reduction pair*)**.** Let $\mathcal{F}$ be a signature and $\mu \in M_{\mathcal{F}}$. A μ-*reduction pair* $(\gtrsim, \sqsupset)$ consists of a stable and μ-monotonic quasi-ordering $\gtrsim$ and a well-founded stable relation $\sqsupset$ on terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$ that are compatible, i.e., $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. We say that $(\gtrsim, \sqsupset)$ is μ-monotonic if $\sqsupset$ is μ-monotonic.

The following result allows us to use a μ-*monotonic* μ-reduction pair to remove some rewrite rules from the original rewrite system $\mathcal{R}$ before starting a termination proof.

Proposition 11 (Removing strict rewrite rules)**.** *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $(\gtrsim, \sqsupset)$ be a μ-monotonic μ-reduction pair such that $l\ (\gtrsim \cup \sqsupset)\ r$ for all $l \to r \in R$. Let $\mathcal{R}_{\sqsupset} = \{l \to r \in R \mid l \sqsupset r\}$ and $\mathcal{S} = \mathcal{R} - \mathcal{R}_{\sqsupset}$. Then, $\mathcal{R}$ is μ-terminating if and only if $\mathcal{S}$ is μ-terminating.*

Proof. Since $\mathcal{S} \subseteq \mathcal{R}$, the *only if* part is obvious. For the *if* part, we proceed by contradiction. If $\mathcal{R}$ is not μ-terminating, then there is an infinite μ-rewrite sequence A:

$$t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \cdots t_n \hookrightarrow_{\mathcal{R},\mu} \cdots$$

where an infinite number of rules in $\mathcal{R}_\sqsupset$ have been used; otherwise, there would be an infinite tail $t_m \hookrightarrow_{\mathcal{S},\mu} t_{m+1} \hookrightarrow_{\mathcal{S},\mu} \cdots$ for some $m \geqslant 1$ where only rules in $\mathcal{S}$ are applied, contradicting the μ-termination of $\mathcal{S}$. Let $J = \{j_1, j_2, \ldots\}$ be the infinite set of indices indicating μ-rewrite steps $t_j \hookrightarrow_{\mathcal{R},\mu} t_{j+1}$ in A, for all $j \in J$, where rules in $\mathcal{R}_\sqsupset$ have been used to perform the μ-rewriting step. Since $l \sqsupset r$ for all $l \to r \in \mathcal{R}_\sqsupset$, by stability and μ-monotonicity of $\sqsupset$, we have that $t_{j_i} \sqsupset t_{j_i+1}$. Since $l \gtrsim r$ for all $l \to r \in S$, by stability and μ-monotonicity of $\gtrsim$, we have that $t_{j_i+1} \gtrsim t_{j_i+1}$. By compatibility between $\gtrsim$ and $\sqsupset$, we have $t_{j_i} \sqsupset t_{j_{i+1}}$ for all $i \geqslant 1$. We obtain an infinite sequence $t_{j_1} \sqsupset t_{j_2} \sqsupset \cdots$ which contradicts well-foundedness of $\sqsupset$. □

10.1. Argument filterings for CSR

An argument filtering π for a signature $\mathcal{F}$ is a mapping that assigns to every k-ary function symbol $f \in \mathcal{F}$ an argument position $i \in \{1, \ldots, k\}$ or a (possibly empty) list $[i_1, \ldots, i_m]$ of argument positions with $1 \leqslant i_1 < \cdots < i_m \leqslant k$ [42]. The *trivial* argument filtering $\pi_\top$ is given by $\pi_\top(f) = [1, \ldots, k]$ for each k-ary symbol $f \in \mathcal{F}$. It corresponds to the argument filtering which does nothing. In the dependency pair method, argument filterings π provide a simple way to *remove* parts of the syntactic structure of a rule $s \to t$. Argument filterings (recursively) drop immediate subterms of terms and can produce terms from a new signature where the arity of symbols has been decreased if necessary. In this way, we obtain simpler expressions that are (hopefully) easy to compare. In the following, we adapt the argument filtering technique to our CSDP framework. In Section 10.2, we investigate their use together with μ-reduction pairs. We can use an argument filtering π to 'filter' either the signature $\mathcal{F}$ or any replacement map $\mu \in M_\mathcal{F}$. In the following, we assume that:

1. The signature $\mathcal{F}_\pi$ consists of all function symbols f such that $\pi(f)$ is some list $[i_1, \ldots, i_m]$, where, in $\mathcal{F}_\pi$, the arity of f is m. As usual, we give the same name to the version of $f \in \mathcal{F}$ that belongs to $\mathcal{F}_\pi$.
2. The replacement map $\mu_\pi \in M_{\mathcal{F}_\pi}$ is given as follows: for all $f \in \mathcal{F}$ such that $f \in \mathcal{F}_\pi$ and $\pi(f) = [i_1, \ldots, i_m]$:

$$\mu_\pi(f) = \{j \in \{1, \ldots, m\} \mid i_j \in \mu(f)\}$$

An argument filtering π induces a mapping from $\mathcal{T}(\mathcal{F}, \mathcal{X})$ to $\mathcal{T}(\mathcal{F}_\pi, \mathcal{X})$, also denoted by π:

$$\pi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \pi(t_i) & \text{if } t = f(t_1, \ldots, t_k) \text{ and } \pi(f) = i \\ f(\pi(t_{i_1}), \ldots, \pi(t_{i_m})) & \text{if } t = f(t_1, \ldots, t_k) \text{ and } \pi(f) = [i_1, \ldots, i_m] \end{cases}$$

Note that, for the *trivial* argument filtering $\pi_\top$, we have that $\mathcal{F}_{\pi_\top} = \mathcal{F}$ and $\mu_{\pi_\top} = \mu$ for all $\mu \in M_\mathcal{F}$. Furthermore, $\pi_\top(t) = t$ for all $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. In the following, given a substitution σ and an argument filtering π, we let σ_π be the substitution defined by $\sigma_\pi(x) = \pi(\sigma(x))$ for all $x \in \mathcal{X}$. The following auxiliary results are used below.

Lemma 7. *Let $\mathcal{F}$ be a signature, π be an argument filtering for $\mathcal{F}$, and σ be a substitution. If $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, then $\pi(\sigma(t)) = \sigma_\pi(\pi(t))$.*

Proof. By structural induction.

1. Base case: t is a variable or a constant symbol. If $t = x \in \mathcal{X}$, then $\pi(x) = x$ and $\pi(\sigma(x)) = \sigma_\pi(x) = \sigma_\pi(\pi(x))$. If t is a constant symbol, then $\pi(t) = t$ and $\sigma(t) = t = \sigma_\pi(t)$. Hence, $\pi(\sigma(t)) = \pi(t) = t = \sigma_\pi(t) = \sigma_\pi(\pi(t))$.
2. If $t = f(t_1, \ldots, t_k)$, then we consider the two possible cases according to $\pi(f)$:
   (a) If $\pi(f) = i$ for some $i \in \{1, \ldots, k\}$, then $\pi(t) = \pi(t_i)$. By the induction hypothesis, $\pi(\sigma(t_i)) = \sigma_\pi(\pi(t_i))$. Therefore, $\pi(\sigma(t)) = \pi(f(\sigma(t_1), \ldots, \sigma(t_k))) = \pi(\sigma(t_i)) = \sigma_\pi(\pi(t_i)) = \sigma_\pi(\pi(f(t_1, \ldots, t_k))) = \sigma_\pi(\pi(t))$.
   (b) If $\pi(f) = [i_1, \ldots, i_m]$, then $\pi(t) = f(\pi(t_{i_1}), \ldots, \pi(t_{i_m}))$. By the induction hypothesis, $\pi(\sigma(t_{i_j})) = \sigma_\pi(\pi(t_{i_j}))$ for all $j \in \{1, \ldots, m\}$. Thus, $\pi(\sigma(t)) = \pi(f(\sigma(t_1), \ldots, \sigma(t_k))) = f(\pi(\sigma(t_{i_1})), \ldots, \pi(\sigma(t_{i_m}))) = f(\sigma_\pi(\pi(t_{i_1})), \ldots, \sigma_\pi(\pi(t_{i_m}))) = \sigma_\pi(f(\pi(t_{i_1}), \ldots, \pi(t_{i_m}))) = \sigma_\pi(\pi(t))$. □

Proposition 12. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_\mathcal{F}$, π be an argument filtering for $\mathcal{F}$, and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. Let $\gtrsim$ be a μ_π-monotonic quasi-ordering such that $\pi(l) \gtrsim \pi(r)$ for all $l \to r \in R$. If $s \hookrightarrow^* t$, then $\pi(s) \gtrsim \pi(t)$.*

Proof. By induction on the length n of the μ-rewrite sequence.

1. If $n = 0$, then $s = t$ and, trivially, $\pi(s) = \pi(t)$. Since $\gtrsim$ is reflexive, we have $\pi(s) \gtrsim \pi(t)$.
2. If $n > 0$, we can write $s \hookrightarrow s' \hookrightarrow^* t$, where the length of the sequence from s' to t is $n - 1$. Let $p \in \mathcal{P}os^\mu(s)$ be the μ-replacing position where the μ-rewriting step $s \hookrightarrow s'$ is performed. We prove that $s \hookrightarrow s'$ implies $\pi(s) \gtrsim \pi(s')$ by induction on the structure of p.

(a) If $p = \Lambda$, then $s = \sigma(l)$ and $s' = \sigma(r)$ for some rewrite rule $l \to r$ and matching substitution σ. By Lemma 7, $\pi(s) = \pi(\sigma(l)) = \sigma_\pi(\pi(l))$ and $\pi(s') = \pi(\sigma(r)) = \sigma_\pi(\pi(r))$. Since $\pi(l) \gtrsim \pi(r)$, by stability of $\gtrsim$ we conclude $\pi(s) = \sigma_\pi(\pi(l)) \gtrsim \sigma_\pi(\pi(r)) = \pi(s')$.

(b) If $p = i \cdot q$, then we can write $s = f(s_1, \ldots, s_i, \ldots, s_k)$ and $s' = f(s'_1, \ldots, s'_i, \ldots, s'_k)$ for some nonconstant symbol f (i.e., $ar(f) > 0$) and we know that $i \in \mu(f)$, $s_i \hookrightarrow s'_i$ at position q, and $s_j = s'_j$ for all $j \neq i$. By the induction hypothesis, $\pi(s_i) \gtrsim \pi(s'_i)$. We consider the two possible cases according to $\pi(f)$:

i. If $\pi(f) = j$ for some $j \in \{1, \ldots, k\}$, then $\pi(s) = \pi(s_j)$. If $i \neq j$, then $s'_j = s_j$. By reflexivity of $\gtrsim$, we have $\pi(s_j) \gtrsim \pi(s'_j)$. If $i = j$, then we know from above that $\pi(s_i) \gtrsim \pi(s'_i)$. Therefore, $\pi(s) = \pi(s_j) \gtrsim \pi(s'_j) = \pi(s')$.

ii. If $\pi(f) = [i_1, \ldots, i_m]$, then we have that $\pi(s) = f(\pi(s_{i_1}), \ldots, \pi(s_{i_m}))$ and $\pi(s') = f(\pi(s'_{i_1}), \ldots, \pi(s'_{i_m}))$. Consider i_j for some $j \in \{1, \ldots, m\}$. We have two cases:

A. If $i_j = i$, then by the induction hypothesis, $\pi(s_{i_j}) \gtrsim \pi(s'_{i_j})$ and, by definition of μ_π, $j \in \mu_\pi(f)$.

B. If $i_j \neq i$, then $s'_{i_j} = s_{i_j}$ and we have $\pi(s_{i_j}) = \pi(s'_{i_j})$.

Note that $\pi(s_{i_j})$ is the jth immediate subterm of $\pi(s)$. By μ_π-monotonicity of $\gtrsim$,

$$\begin{aligned}
\pi(s) &= \pi(f(s_1, \ldots, s_k)) \\
&= f(\pi(s_{i_1}), \ldots, \pi(s_{i_j}), \ldots, \pi(s_{i_m})) \\
&\gtrsim f(\pi(s_{i_1}), \ldots, \pi(s'_{i_j}), \ldots, \pi(s_{i_m})) \\
&= f(\pi(s'_{i_1}), \ldots, \pi(s'_{i_j}), \ldots, \pi(s'_{i_m})) \\
&= \pi(f(s'_1, \ldots, s'_k)) \\
&= \pi(s')
\end{aligned}$$

where we assume that $i_j = i$ for some $j \in \{1, \ldots, k\}$. If no such j exists, then we would have $\pi(s) = \pi(s')$, which also implies $\pi(s) \gtrsim (s')$ because $\gtrsim$ is reflexive.

Thus, we have proved that $s \hookrightarrow s'$ implies $\pi(s) \gtrsim \pi(s')$ as desired.

Therefore, $\pi(s) \gtrsim \pi(s')$ and, by the induction hypothesis, $\pi(s') \gtrsim \pi(t)$. By transitivity of $\gtrsim$, we conclude $\pi(s) \gtrsim \pi(t)$. □

Remark 12. We often use argument filterings to transform (sets of) rules S as follows: $\pi(s \to t) = \pi(s) \to \pi(t)$ for a rule $s \to t$, and $\pi(S) = \{\pi(s \to t) \mid s \to t \in S\}$. Given a TRS $\mathcal{R} = (\mathcal{F}, R)$, we write $\pi(\mathcal{R})$ to denote the filtered TRS $(\mathcal{F}_\pi, \pi(R))$.

10.2. Removing pairs using μ-reduction pairs

Given TRSs $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$, and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$, checking the absence of infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chains can often be 'simplified' to checking the absence of infinite minimal $(\mathcal{P}', \mathcal{R}, \mu)$-chains for a proper subTRS $\mathcal{P}'$ of $\mathcal{P}$ by finding appropriate *μ-reduction pairs* $(\gtrsim, \sqsupset)$. The presence of *collapsing* pairs $u \to v = u \to x \in \mathcal{P}_\mathcal{X}$ imposes some additional requirements on the μ-reduction pairs:

1. We need to ensure that the quasi-ordering $\gtrsim$ is able to 'look' for a μ-replacing subterm s inside the instantiation $\sigma(x)$ of a migrating variable x: since $\sigma(x) = C[s]_p$ for some context $C[\ ]_p$ and μ-replacing position $p \in \mathcal{P}os^\mu(C[\ ]_p)$ such that $\mathrm{sprefix}_{C[\ ]_p}(p) \subseteq \mathcal{F}$, we can obtain s out from $C[s]_p$ by applying the projection rules in $\mathcal{E}mb^\mu(\mathcal{F})$ (Definition 1). Hence, we require $\mathcal{E}mb^\mu(\mathcal{F}) \subseteq \gtrsim$.
2. We need to connect the marked version $s^\sharp$ of s (which is an instance of a hidden term $t \in \mathcal{NHT}_\mathcal{P}$, i.e., $s = \theta(t)$ for some substitution θ) with an instance $\sigma(u)$ of the left-hand side u of a pair; hence, we require $t \gtrsim t^\sharp$ or $t \sqsupset t^\sharp$ for all $t \in \mathcal{NHT}_\mathcal{P}$ which, by stability, becomes $s \gtrsim s^\sharp$ or $s \sqsupset s^\sharp$.

The following theorem formalizes a generic processor to remove pairs from $\mathcal{P}$ by using argument filterings and μ-reduction pairs.

Theorem 9 (μ-reduction pair processor)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let π be an argument filtering for $\mathcal{F} \cup \mathcal{G}$ and $(\gtrsim, \sqsupset)$ be a μ_π-reduction pair such that*

1. *$\pi(\mathcal{R}) \subseteq \gtrsim$, $\pi(\mathcal{P}) \subseteq \gtrsim \cup \sqsupset$, and*
2. *whenever $\mathcal{NHT}_\mathcal{P} \neq \emptyset$ and $\mathcal{P}_\mathcal{X} \neq \emptyset$, we have that*
   (a) *for all $f \in \mathcal{F}$, either $\pi(f) = [i_1, \ldots, i_m]$ and $\mu(f) \subseteq \pi(f)$, or $\pi(f) = i$ and $\mu(f) = \{i\}$,*
   (b) *$\mathcal{E}mb^{\mu_\pi}(\mathcal{F}_\pi) \subseteq \gtrsim$, and*

(c) $\pi(t)\ (\gtrsim \cup \sqsupseteq)\ \pi(t^\sharp)$ *for all* $t \in \mathcal{NHT}_\mathcal{P}$,

Let $\mathcal{P}_\sqsupset = \{u \to v \in \mathcal{P} \mid \pi(u) \sqsupset \pi(v)\}$. *Then, the processor* Proc_{RP} *given by*

$$\mathsf{Proc}_{RP}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_\sqsupset, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. Completeness is obvious, since $\mathcal{P} - \mathcal{P}_\sqsupset \subseteq \mathcal{P}$. Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A, but that there is no infinite minimal $(\mathcal{P} - \mathcal{P}_\sqsupset, \mathcal{R}, \mu)$-chain. Due to the finiteness of $\mathcal{P}$, we assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B

$$\sigma(u_1) \hookrightarrow_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2) \hookrightarrow_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_2 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_3) \hookrightarrow_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu \cdots$$

for some substitution σ, where all pairs in $\mathcal{Q}$ are infinitely often used. Also, for all $i \geqslant 1$, (1) if $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$, then $t_i = \sigma(v_i)$ and (2) if $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_\mathcal{X}$, then $t_i = s_i^\sharp$ for some s_i such that $\sigma(x_i) = C_i[s_i]_{p_i}$ for some $C_i[\ ]_{p_i}$ and $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$ such that $\mathsf{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ and $s_i = \theta_i(\bar{s}_i)$ for some $\bar{s}_i \in \mathcal{NHT}$ and substitution θ_i. Actually, since $t_i = s_i^\sharp = \theta_i(\bar{s}_i)^\sharp = \theta_i(\bar{s}_i^\sharp)$ and $t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$, we can further say that $\bar{s}_i \in \mathcal{NHT}_\mathcal{Q}$.

Since $\pi(u_i)\ (\gtrsim \cup \sqsupseteq)\ \pi(v_i)$ for all $u_i \to v_i \in \mathcal{Q} \subseteq \mathcal{P}$, by stability of $\gtrsim$ and $\sqsupseteq$, we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupseteq)\ \sigma_\pi(\pi(v_i))$ for all $i \geqslant 1$.

No pair $u \to v \in \mathcal{Q}$ satisfies that $\pi(u) \sqsupset \pi(v)$. Otherwise, we get a contradiction by considering the following two cases:

1. If $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$, then $t_i = \sigma(v_i) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$ and by Proposition 12, $\pi(t_i) \gtrsim \pi(\sigma(u_{i+1}))$. By Lemma 7, $\pi(t_i) \gtrsim \sigma_\pi(\pi(u_{i+1}))$. Since we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupseteq)\ \sigma_\pi(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$ (using Lemma 7), by using transitivity of $\gtrsim$ and compatibility between $\gtrsim$ and $\sqsupseteq$, we conclude that $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupseteq)\ \sigma_\pi(\pi(u_{i+1}))$.
2. If $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_\mathcal{X}$, then $\sigma(v_i) = \sigma(x_i) = C_i[s_i]_{p_i}$. Since $i \in \mu(f)$ implies that $i \in \pi(f)$, we can say that $\pi(\sigma(x)) = \sigma_\pi(x) = \pi(C_i)[\pi(s_i)]_{q_i}$ for some $q_i \in \mathcal{P}os^{\mu_\pi}(\pi(C_i))$ and $\mathsf{sprefix}_{\pi(C_i)}(q_i) \subseteq \mathcal{F}_\pi$. Since $\mathcal{E}mb^{\mu_\pi}(\mathcal{F}_\pi) \subseteq \gtrsim$, we have $\sigma_\pi(\pi(v_i)) = \sigma_\pi(x_i) \gtrsim \pi(s_i)$. Furthermore, we are assuming that $\pi(t)\ (\gtrsim \cup \sqsupseteq)\ \pi(t^\sharp)$ for all $t \in \mathcal{NHT}_\mathcal{Q} \subseteq \mathcal{NHT}_\mathcal{P}$. Since $s_i = \theta_i(\bar{s}_i)$, we have that $\pi(s_i) = \pi(\theta_i(\bar{s}_i)) = \theta_{i,\pi}(\pi(\bar{s}_i))$ (using Lemma 7 again) and, similarly, $\pi(s_i^\sharp) = \theta_{i,\pi}(\pi(\bar{s}_i^\sharp))$. By stability we have that $\pi(s_i)\ (\gtrsim \cup \sqsupseteq)\ \pi(s_i^\sharp)$. Hence, by transitivity of $\gtrsim$ (and compatibility of $\gtrsim$ and $\sqsupseteq$), we have $\sigma_\pi(\pi(v_i)) = \sigma_\pi(x_i)\ (\gtrsim \cup \sqsupseteq)\ \pi(s_i^\sharp)$. Finally, since $\pi(s_i^\sharp) = \pi(t_i)$ and $t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$ for all $i \geqslant 1$, by Proposition 12 and Lemma 7, $\pi(t_i) \gtrsim \sigma_\pi(\pi(u_{i+1}))$. Therefore, again by transitivity of $\gtrsim$ and compatibility of $\gtrsim$ and $\sqsupseteq$, we conclude that $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupseteq)\ \sigma_\pi(\pi(u_{i+1}))$.

Since $u \to v$ occurs infinitely often in B, there is an infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that $\sigma_\pi(\pi(u_i)) \sqsupset \sigma_\pi(\pi(u_{i+1}))$ for all $i \in \mathcal{I}$. And we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupseteq)\ \sigma_\pi(\pi(u_{i+1}))$ for all other $u_i \to v_i \in \mathcal{Q}$. Thus, by using the compatibility conditions of the μ_π-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence that contradicts the well-foundedness of $\sqsupset$.

Therefore, $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_\sqsupset$, which means that B is an infinite minimal $(\mathcal{P} - \mathcal{P}_\sqsupset, \mathcal{R}, \mu)$-chain, thus leading to a contradiction. □

Example 18. Consider the TRS $\mathcal{R}$ [63, Example 5]:

$$\mathtt{if}(\mathtt{true}, x, y) \to x \qquad \mathtt{f}(x) \to \mathtt{if}(x, \mathtt{c}, \mathtt{f}(\mathtt{true}))$$
$$\mathtt{if}(\mathtt{false}, x, y) \to y$$

with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{if}) = \{1, 2\}$. Then, $\mathsf{DP}(\mathcal{R}, \mu)$ consists of the following CSDPs:

$$\mathtt{F}(x) \to \mathtt{IF}(x, \mathtt{c}, \mathtt{f}(\mathtt{true})) \qquad \mathtt{IF}(\mathtt{false}, x, y) \to y$$

with $\mu^\sharp(\mathtt{F}) = \{1\}$ and $\mu^\sharp(\mathtt{IF}) = \{1, 2\}$. The μ-reduction pair $(\geqslant, >)$ induced by the polynomial interpretation[6]

$$[\mathtt{c}] = [\mathtt{true}] = 0 \qquad [\mathtt{f}](x) = x \qquad [\mathtt{F}](x) = x$$
$$[\mathtt{false}] = 1 \qquad [\mathtt{if}](x, y, z) = x + y + z \qquad [\mathtt{IF}](x, y, z) = x + z$$

[6] See [49] for more information about the automatic generation of polynomial (quasi-)orderings with monotonicity requirements specified by means of replacement maps.

can be used to prove the μ-termination of $\mathcal{R}$. For $\mathcal{P} = \mathsf{DP}(\mathcal{R}, \mu)$, we have $\mathcal{NHT}_{\mathcal{P}} = \{\mathtt{f(true)}\}$. First, we can see that the quasi-ordering is compatible with the rules in $\mathcal{E}mb^{\mu}(\mathcal{F})$:

$$\begin{aligned}
[\mathtt{f}(x)] &= x &&\geqslant x = [x]\\
[\mathtt{if}(x, y, z)] &= x + y + z &&\geqslant x = [x]\\
[\mathtt{if}(x, y, z)] &= x + y + z &&\geqslant y = [y]
\end{aligned}$$

Now we can see that the condition on the only hidden term in $\mathcal{NHT}_{\mathcal{P}}$ is also fulfilled:

$$[\mathtt{f(true)}] = 0 \geqslant 0 = [\mathtt{F(true)}]$$

Finally, for the three rules in $\mathcal{R}$ and the two pairs in $\mathcal{P}$, we have:

$$\begin{aligned}
[\mathtt{f}(x)] &= x &&\geqslant x = [\mathtt{if}(x, \mathtt{c}, \mathtt{f(true)})]\\
[\mathtt{if(true}, x, y)] &= x + y &&\geqslant x = [x]\\
[\mathtt{if(false}, x, y)] &= x + y + 1 &&\geqslant y = [y]\\
[\mathtt{F}(x)] &= x &&\geqslant x = [\mathtt{IF}(x, \mathtt{c}, \mathtt{f(true)})]\\
[\mathtt{IF(false}, x, y)] &= y + 1 &&> y = [y]
\end{aligned}$$

We remove the 'strict' pair $\mathtt{IF(false}, x, y) \to y$ from $\mathcal{P}$ to obtain $\mathcal{P}'$. With $(\mathcal{P}', \mathcal{R}, \mu^{\sharp})$, the application of Proc_{SCC} leads to an empty set of CS problems. Thus, the μ-termination of $\mathcal{R}$ is proved.

The 'compatibility' between the replacement map μ and the argument filtering π, which is required when collapsing pairs are present, is necessary in Theorem 9.

Example 19. Consider the following TRS $\mathcal{R}$:

$$\begin{aligned}
\mathtt{a} &\to \mathtt{c(h(f(a), b))}\\
\mathtt{f(c}(x)) &\to x
\end{aligned}$$

together with the replacement map μ given by $\mu(\mathtt{f}) = \mu(\mathtt{h}) = \{1\}$ and $\mu(\mathtt{c}) = \varnothing$. $\mathsf{DP}(\mathcal{R}, \mu)$ is:

$$\mathtt{F(c}(x)) \to x$$

and $\mathcal{NHT}_{\mathsf{DP}(\mathcal{R},\mu)} = \{\mathtt{f}(\mathtt{a})\}$. Note that $\mathcal{R}$ is *not* μ-terminating:

$$\mathtt{f}(\underline{\mathtt{a}}) \hookrightarrow \underline{\mathtt{f(c(h(f(a), b)))}} \hookrightarrow \mathtt{h(f}(\underline{\mathtt{a}}), \mathtt{b}) \hookrightarrow \cdots$$

For the argument filtering π given by $\pi(\mathtt{a}) = \pi(\mathtt{h}) = []$, $\pi(\mathtt{F}) = \pi(\mathtt{f}) = [1]$ and $\pi(\mathtt{c}) = 1$, $\mathcal{F}_{\pi}$ consists of the constants $\mathtt{a}$, $\mathtt{h}$ and symbol $\mathtt{f}$ of arity 1. Also, $\mu^{\sharp}_{\pi}(\mathtt{f}) = \mu^{\sharp}_{\pi}(\mathtt{F}) = \{1\}$ and $\mu^{\sharp}_{\pi}(\mathtt{a}) = \mu^{\sharp}_{\pi}(\mathtt{h}) = \varnothing$. We get the constraints:

$$\begin{aligned}
\pi(\mathtt{a}) &= \mathtt{a} &&\gtrsim \mathtt{h} &&= \pi(\mathtt{c(h(f(a), b))})\\
\pi(\mathtt{f(c}(x))) &= \mathtt{f}(x) &&\gtrsim x &&= \pi(x)\\
&\quad \mathtt{f}(x) &&\gtrsim x\\
\pi(\mathtt{f(a)}) &= \mathtt{f(a)} &&\gtrsim \mathtt{F(a)} &&= \pi(\mathtt{F(a)})\\
\pi(\mathtt{F(c}(x))) &= \mathtt{F}(x) &&\sqsupset x &&= \pi(x)
\end{aligned}$$

which are easily satisfiable (by a polynomial interpretation, for instance). We would wrongly conclude μ-termination of $\mathcal{R}$. Note that $\pi(\mathtt{c}) = 1$ but $\mu^{\sharp}(\mathtt{c}) = \varnothing$, and that $\pi(\mathtt{h}) = []$ but $\mu^{\sharp}(\mathtt{h}) = \{1\}$.

The next processor is useful when all (filterings of) terms in $\mathcal{NHT}_{\mathcal{P}}$ are *ground*. The advantage is that the quasi-ordering $\gtrsim$ of the μ-reduction pair does not need to impose compatibility with the rules in $\mathcal{E}mb^{\mu}(\mathcal{F})$.

Theorem 10 (μ-reduction pair processor for ground hidden terms)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let π be an argument filtering for $\mathcal{F} \cup \mathcal{G}$ such that, for all $t \in \mathcal{NHT}_{\mathcal{P}}$, $\pi(t)$ is ground. Let $(\gtrsim, \sqsupset)$ be a μ_{π}-reduction pair such that*

1. $\pi(\mathcal{R}) \subseteq \gtrsim, \pi(\mathcal{P}_\mathcal{G}) \subseteq \gtrsim \cup \sqsupset$, *and*
2. *for all* $u \to v \in \mathcal{P}_\mathcal{X}$ *and all* $t \in \mathcal{NHT}_\mathcal{P}$, $\pi(u)\ (\gtrsim \cup \sqsupset)\ \pi(t^\sharp)$

Let $\mathcal{P}_\sqsupset = \{u \to v \in \mathcal{P}_\mathcal{G} \mid \pi(u) \sqsupset \pi(v)\} \cup \{u \to v \in \mathcal{P}_\mathcal{X} \mid \forall t \in \mathcal{NHT}_\mathcal{P}, \pi(u) \sqsupset \pi(t^\sharp)\}$. *Then, the processor* Proc_{RPg} *given by*

$$\mathsf{Proc}_{RPg}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_\sqsupset, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. The proof is analogous to that of Theorem 9. Assume the facts and notation in the first paragraph of such a proof. Again, we proceed by contradiction and assume that a pair $u \to v \in \mathcal{Q}$ is in $\mathcal{P}_\sqsupset$. Again, we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupset)\ \sigma_\pi(\pi(u_{i+1}))$ for all pairs $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$.

Now, if $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_\mathcal{X}$, then since $\pi(u_i)\ (\gtrsim \cup \sqsupset)\ \pi(t^\sharp)$ for all $t \in \mathcal{NHT}_\mathcal{Q} \subseteq \mathcal{NHT}_\mathcal{P}$, by stability we have that $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupset)\ \sigma_\pi(\pi(t^\sharp))$. Since $\pi(t)$ is ground, we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupset)\ \pi(t^\sharp)$. Therefore, since $s_i \in \mathcal{NHT}_\mathcal{Q}$ and $t_i = s_i^\sharp$, we have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupset)\ \pi(t_i)$. Finally, since $s_i^\sharp = t_i$ and $t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$ for all $i \geqslant 1$, by Proposition 12 and Lemma 7, we have that $\pi(t_i) \gtrsim \sigma_\pi(\pi(u_{i+1}))$. Thus, we also have $\sigma_\pi(\pi(u_i))\ (\gtrsim \cup \sqsupset)\ \sigma_\pi(\pi(u_{i+1}))$.

Since $u \to v$ occurs infinitely often in B, by using the compatibility conditions of the μ_π-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence that contradicts well-foundedness of $\sqsupset$. In particular, if $u \to v \in \mathcal{Q}_\mathcal{X} \cap \mathcal{P}_\sqsupset$, then $\pi(u) \sqsupset \pi(t^\sharp)$ for all $t \in \mathcal{NHT}_\mathcal{Q}$, so each time that $u \to v$ is used, a strict decrease occurs. □

Theorem 10 can succeed when Theorem 9 fails.

Example 20. Consider the TRS $\mathcal{R}$:

$$\mathtt{a} \to \mathtt{f}(\mathtt{d}(\mathtt{c}(\mathtt{a}))) \tag{27}$$

$$\mathtt{f}(\mathtt{c}(x)) \to x \tag{28}$$

$$\mathtt{d}(\mathtt{c}(x)) \to \mathtt{b} \tag{29}$$

and the replacement map μ given by $\mu(\mathtt{c}) = \varnothing$ and $\mu(\mathtt{f}) = \mu(\mathtt{d}) = \{1\}$. There are three CSDPs:

$$\mathtt{A} \to \mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a}))) \tag{30}$$

$$\mathtt{A} \to \mathtt{D}(\mathtt{c}(\mathtt{a})) \tag{31}$$

$$\mathtt{F}(\mathtt{c}(x)) \to x \tag{32}$$

$\mathsf{Proc}_{SCC}(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ yields a single CS problem $(\mathcal{P}, \mathcal{R}, \mu)$ with $\mathcal{P} = \{(30), (32)\}$. Since $\mathcal{NHT}_\mathcal{P} = \{\mathtt{a}\} \neq \varnothing$ and $\mathtt{F}(\mathtt{c}(x)) \to x$ is a collapsing CSDP, according to Theorem 9 we would require that any μ-reduction ordering used in the theorem satisfy $\mathcal{E}mb^\mu(\mathcal{F}) \subseteq \gtrsim$ (assume the trivial filtering $\pi_\top$ here) and that $\mathtt{a}\ (\gtrsim \cup \sqsupset)\ \mathtt{A}$. In this case, though, since $\mathtt{d}(\mathtt{c}(\mathtt{a})) \trianglerighteq_\mu \mathtt{c}(\mathtt{a})$, we must have $\mathtt{d}(\mathtt{c}(\mathtt{a})) \gtrsim \mathtt{c}(\mathtt{a})$; by μ-monotonicity of $\gtrsim$, $\mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a}))) \gtrsim \mathtt{F}(\mathtt{c}(\mathtt{a}))$. Now, one of the following two cases must hold:

1. $\mathtt{A} \sqsupset \mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a})))$ and $\mathtt{F}(\mathtt{c}(x))\ (\gtrsim \cup \sqsupset)\ x$. By stability of $\gtrsim$ and $\sqsupset$, we have $\mathtt{F}(\mathtt{c}(\mathtt{a}))\ (\gtrsim \cup \sqsupset)\ \mathtt{a}$. Thus,

   $$\mathtt{A} \sqsupset \mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a}))) \gtrsim \mathtt{F}(\mathtt{c}(\mathtt{a}))\ (\gtrsim \cup \sqsupset)\ \mathtt{a}\ (\gtrsim \cup \sqsupset)\ \mathtt{A}.$$

   By compatibility of $\gtrsim$ and $\sqsupset$, we have $\mathtt{A} \sqsupset \cdots \sqsupset \mathtt{A}$, contradicting the well-foundedness of $\sqsupset$.
2. $\mathtt{A}\ (\gtrsim \cup \sqsupset)\ \mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a})))$ and $\mathtt{F}(\mathtt{c}(x)) \sqsupset x$. Hence,

   $$\mathtt{A}\ (\gtrsim \cup \sqsupset)\ \mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a}))) \gtrsim \mathtt{F}(\mathtt{c}(\mathtt{a})) \sqsupset \mathtt{a}\ (\gtrsim \cup \sqsupset)\ \mathtt{A}.$$

   Again, by compatibility of $\gtrsim$ and $\sqsupset$, we have $\mathtt{A} \sqsupset \cdots \sqsupset \mathtt{A}$.

Thus, Theorem 9 cannot be used with this example. Since $\mathcal{NHT}_\mathcal{P} \subseteq \mathcal{T}(\mathcal{F})$, Theorem 10 is applicable here. The μ-reduction pair $(\geqslant, >)$ induced by the following polynomial interpretation: [7]

[7] See [49,51] for details about the use of polynomial intepretations with rational coefficients.

$$[\mathtt{a}] = 1 \qquad [\mathtt{b}] = 0 \qquad [\mathtt{c}](x) = x+1 \qquad [\mathtt{d}](x) = \tfrac{1}{4}x$$
$$[\mathtt{f}](x) = x \qquad [\mathtt{A}] = 1 \qquad [\mathtt{F}](x) = 0$$

can be used to remove (30) from $\mathcal{P}$. For the three rules in $\mathcal{R}$ and the two pairs in $\mathcal{P}$, we have:

$$\begin{aligned}
[\mathtt{a}] &= 1 &&\geqslant \tfrac{1}{2} = [\mathtt{f}(\mathtt{d}(\mathtt{c}(\mathtt{a})))]\\
[\mathtt{f}(\mathtt{c}(x))] &= x+1 &&\geqslant x = [x]\\
[\mathtt{d}(\mathtt{c}(x))] &= \tfrac{1}{4}x+\tfrac{1}{4} &&\geqslant 0 = [\mathtt{b}]\\
[\mathtt{A}] &= 1 &&> \tfrac{1}{2} = [\mathtt{F}(\mathtt{d}(\mathtt{c}(\mathtt{a})))]\\
[\mathtt{F}(\mathtt{c}(x))] &= x+1 &&\geqslant 1 = [\mathtt{A}]
\end{aligned}$$

We remove (30) from $\mathcal{P}$ to obtain $\mathcal{P}' = \{(32)\}$. Now, $\mathsf{Proc}_{SCC}(\mathcal{P}', \mathcal{R}, \mu) = \varnothing$ because $\mathcal{NHT}_{\mathcal{P}'} = \varnothing$ and $\mathsf{EG}(\mathcal{P}', \mathcal{R}, \mu)$ contains no cycle. Thus, the μ-termination of $\mathcal{R}$ is proved.

Nevertheless, even with $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$, Theorem 9 can be helpful when Theorem 10 fails.

Example 21. Consider $\mathcal{R}$ and μ as in Example 16. Theorem 10 cannot be used here because, reasoning as in Example 16, we would obtain constraints that are incompatible with the well-foundedness of $\sqsupset$ for any strict component $\sqsupset$ of a μ-reduction pair $(\gtrsim, \sqsupset)$. However, the μ-termination of $\mathcal{R}$ can be easily proved with Theorem 9. The μ-reduction pair $(\geqslant, >)$ generated by the following polynomial interpretation:

$$[\mathtt{b}] = 1 \qquad [\mathtt{c}](x) = 0 \qquad [\mathtt{f}](x) = x$$
$$[\mathtt{B}] = 2 \qquad [\mathtt{F}](x) = x+1$$

satisfies the requirements of Theorem 10 and can be used to show a weak decrease of the rules and a strict decrease of the two CSDPs which can both be removed.

Our last result establishes that if we are able to provide a strict comparison between unmarked and marked versions of the (filtered) hidden terms in $\mathcal{NHT}_{\mathcal{P}}$, then we can remove *all* collapsing pairs at the same time.

Theorem 11 (μ-reduction pair processor for collapsing pairs)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let π be an argument filtering for $\mathcal{F} \cup \mathcal{G}$ and $(\gtrsim, \sqsupset)$ be a μ_π-reduction pair such that*

1. $\pi(\mathcal{R}) \subseteq \gtrsim, \pi(\mathcal{P}) \subseteq \gtrsim \cup \sqsupset$,
2. $\pi(t) \sqsupset \pi(t^\sharp)$ *for all* $t \in \mathcal{NHT}_{\mathcal{P}}$ *and*
   (a) *for all* $f \in \mathcal{F}$, *either* $\pi(f) = [i_1, \ldots, i_m]$ *and* $\mu(f) \subseteq \pi(f)$, *or* $\pi(f) = i$ *and* $\mu(f) = \{i\}$,
   (b) $\mathcal{E}mb^{\mu_\pi}(\mathcal{F}_\pi) \subseteq \gtrsim$.

Then, the processor Proc_{RPc} *given by*

$$\mathsf{Proc}_{RPc}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold}\\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise}\end{cases}$$

is sound and complete.

Proof. As in the proof of Theorem 9, we proceed by contradiction. We assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A, but that there is no infinite minimal $(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)$-chain. Thus, there is $\mathcal{Q} \subseteq \mathcal{P}$ such that $\mathcal{Q} \cap \mathcal{P}_\mathcal{X} \neq \varnothing$ and A has a tail B as in the proof of Theorem 9. Now, we assume the notation as in the first paragraph of such a proof.

We have $\sigma_\pi(\pi(u_i))$ $(\gtrsim \cup \sqsupset)$ $\pi(t_i)$ and $\pi(t_i) \gtrsim \sigma_\pi(\pi(u_{i+1}))$ for all pairs $u_i \to v_i \in \mathcal{P}_\mathcal{G}$. If $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_\mathcal{X}$, then by applying the considerations in the corresponding item of the proof of Theorem 9 and taking into account that $\pi(t) \sqsupset \pi(t^\sharp)$ for *all* $t \in \mathcal{NHT}_{\mathcal{P}}$, we now have that $\sigma_\pi(\pi(u_i))$ $(\gtrsim \cup \sqsupset)$ $\sigma_\pi(x_i) \sqsupset \pi(t_i) \gtrsim \sigma_\pi(\pi(u_{i+1}))$. Since pairs $u_i \to v_i \in \mathcal{Q}_\mathcal{X}$ occur infinitely often in B, by using the compatibility conditions of the μ_π-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence that contradicts the well-foundedness of $\sqsupset$. □

11. Subterm criterion

In [38], Hirokawa and Middeldorp introduce a *subterm criterion* that permits certain cycles of the dependency graph to be ignored *without paying attention to the rules of the TRS*. Their result applies to *cycles* in the *dependency graph*. Thiemann has adapted it to the DP-framework [62, Section 4.6]. In our adaptation to *CSR*, we take ideas from both works. Our first definition is inspired by Thiemann's *head symbols* [62, Definition 4.36].

Definition 12 (*Root symbols of a TRS*)**.** Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS. The set of *root symbols* associated to $\mathcal{R}$ is:

$$Root(\mathcal{R}) = \{root(l) \mid l \to r \in R\} \cup \{root(r) \mid l \to r \in R, r \notin \mathcal{X}\}$$

The following result relates $Root(\mathcal{P})$ and the set $\mathcal{H}_\mathcal{P}$ of hidden symbols occurring *at the root* of terms in $\mathcal{NHT}_\mathcal{P}(\mathcal{R}, \mu)$. It is silently used in the statements of some theorems below.

Lemma 8. *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs such that $Root(\mathcal{P}) \cap \mathcal{D} = \emptyset$, and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. For all $f \in \mathcal{H}_\mathcal{P}$, we have $f^\sharp \in Root(\mathcal{P})$.*

Proof. If $f \in \mathcal{H}_\mathcal{P}$, then there is $t \in \mathcal{NHT}_\mathcal{P}$ such that $f = root(t)$. Therefore, there are substitutions θ and θ' such that $\theta(t^\sharp) \hookrightarrow^*_{\mathcal{R},\mu} \theta'(u)$ for some $u \to v \in \mathcal{P}$. Since $f^\sharp \notin \mathcal{F}$, μ-rewritings on $\theta(t^\sharp)$ using $\mathcal{R}$ do not remove it. Thus, $root(u) = f^\sharp$ and $f^\sharp \in Root(\mathcal{P})$. □

Thiemann uses argument filterings (see Section 10.1) instead of *simple projections* [38, Definition 10]. We find it more convenient to follow Hirokawa and Middeldorp's style, so we generalize their definition to be used with TRSs rather than cycles in the dependency graph.

Definition 13 (*Simple projection*)**.** Let $\mathcal{R}$ be a TRS. A *simple projection* for $\mathcal{R}$ is a mapping π that assigns to every k-ary symbol $f \in Root(\mathcal{R})$ an argument position $i \in \{1, \ldots, k\}$. The mapping that assigns a subterm $\pi(t) = t|_{\pi(f)}$ to every term $t = f(t_1, \ldots, t_k)$ with $f \in Root(\mathcal{R})$ is also denoted by π; we also let $\pi(x) = x$ if $x \in \mathcal{X}$.

Given a simple projection π for a TRS $\mathcal{R}$, we let $\pi(\mathcal{R}) = \{\pi(l) \to \pi(r) \mid l \to r \in \mathcal{R}\}$.

Theorem 12 (Subterm processor for noncollapsing pairs)**.** *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs such that $\mathcal{P}$ contains no collapsing rule, i.e., for all $u \to v \in \mathcal{P}$, $v \notin \mathcal{X}$, and $Root(\mathcal{P}) \cap \mathcal{D} = \emptyset$. Let $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$ and let π be a simple projection for $\mathcal{P}$. Let $\mathcal{P}_{\pi,\rhd_\mu} = \{u \to v \in \mathcal{P} \mid \pi(u) \rhd_\mu \pi(v)\}$. Then, the processor* $\mathsf{Proc}_{subNColl}$ *given by*

$$\mathsf{Proc}_{subNColl}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_{\pi,\rhd_\mu}, \mathcal{R}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \trianglerighteq_\mu \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. Completeness is obvious because $\mathcal{P} - \mathcal{P}_{\pi,\rhd_\mu} \subseteq \mathcal{P}$. For soundness, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A but there is no infinite minimal $(\mathcal{P} - \mathcal{P}_{\pi,\rhd_\mu}, \mathcal{R}, \mu)$-chain. Since $\mathcal{P}$ is finite, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B that is an infinite minimal $(\mathcal{Q}, \mathcal{R}, \mu)$-chain where all pairs in $\mathcal{Q}$ are infinitely often used. Assume that B is as follows (since $\mathcal{Q}_\mathcal{X} = \emptyset$, we use a simpler notation):

$$t_0 \hookrightarrow^*_{\mathcal{R},\mu} s_1 \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} t_1 \hookrightarrow^*_{\mathcal{R},\mu} s_2 \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} t_2 \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

where there is a substitution σ such that, for all $i \geqslant 1$, $s_i = \sigma(u_i)$ and $t_i = \sigma(v_i)$ for some $u_i \to v_i \in \mathcal{Q}$. Furthermore, w.l.o.g. we also assume that $t_0 = \sigma(v_0)$ for some $u_0 \to v_0 \in \mathcal{P}$.

Note that, for all $i \geqslant 1$, $root(s_i) \in Root(\mathcal{P})$ because $root(u_i) \in Root(\mathcal{P})$. Since $root(v_i) \notin \mathcal{X}$, we have that $root(v_i) \in Root(\mathcal{P})$. Then, for all $i \geqslant 0$, $root(t_i) \in Root(\mathcal{P})$. Therefore, we can apply π to s_{i+1} and t_i for all $i \geqslant 0$. Moreover, since $t_i \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}$ for all $i \geqslant 0$ and $Root(\mathcal{P}) \cap \mathcal{D} = \emptyset$, we can actually write $t_i \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathcal{R},\mu} s_{i+1}$ because μ-rewritings with $\mathcal{R}$ cannot change $root(t_i)$. Hence, $\pi(t_i) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_{i+1})$ and also $root(t_i) = root(s_{i+1})$ for all $i \geqslant 0$. Finally, since $\pi(u_i) \trianglerighteq_\mu \pi(v_i)$ for all $i \geqslant 0$, by stability of $\trianglerighteq_\mu$, we have

$$\pi(s_i) = \pi(\sigma(u_i)) = \sigma(\pi(u_i)) \trianglerighteq_\mu \sigma(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$$

for all $i \geqslant 1$. No pair $u \to v \in \mathcal{Q}$ satisfies that $\pi(u) \rhd_\mu \pi(v)$. Otherwise, we get a contradiction in both of the following two complementary cases:

1. If $\pi(f) \notin \mu(f)$ for all $f \in \mathit{Root}(\mathcal{Q})$, then, for all $i \geqslant 0$, $\pi(t_i) = \pi(s_{i+1})$, because no μ-rewritings are possible on the $\pi(\mathit{root}(t_i))$th immediate subterm $\pi(t_i)$ of t_i. Since $\pi(s_{i+1}) \trianglerighteq_\mu \pi(t_{i+1})$, we have that $\pi(t_i) \trianglerighteq_\mu \pi(t_{i+1})$ for all $i \geqslant 0$. Furthermore, since we assume $\pi(u) \rhd_\mu \pi(v)$ for some $u \to v \in \mathcal{Q}$ which occurs infinitely often in B, and by stability of $\rhd_\mu$, there is a maximal infinite set $J = \{j_1, j_2, \ldots\} \subseteq \mathbb{N}$ such that $\pi(t_{j_i}) \rhd_\mu \pi(t_{j_i+1})$ for all $i \geqslant 1$. We obtain an infinite sequence $\pi(t_{j_1}) \rhd_\mu \pi(t_{j_2}) \rhd_\mu \cdots$ which contradicts the well-foundedness of $\rhd_\mu$.
2. If $\pi(f) \in \mu(f)$ for some $f \in \mathit{Root}(\mathcal{Q})$, then, since $\mathit{root}(t_i) = \mathit{root}(s_{i+1})$ and all pairs in $\mathcal{Q}$ occur infinitely often in B, we can assume that $\mathit{root}(t_0) = f$. Furthermore, since A is minimal, we can assume that t_0 is μ-terminating (with respect to $\mathcal{R}$). Since $\pi(t_i) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_{i+1})$ and $\pi(s_{i+1}) \trianglerighteq_\mu \pi(t_{i+1})$ for all $i \geqslant 0$, the sequence B is transformed into an infinite $\hookrightarrow_{\mathcal{R},\mu} \cup \trianglerighteq_\mu$-sequence

$$\pi(t_0) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_1) \trianglerighteq_\mu \pi(t_1) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_2) \trianglerighteq_\mu \pi(t_2) \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

containing infinitely many $\rhd_\mu$-steps, due to $\pi(u) \rhd_\mu \pi(v)$ for some $u \to v \in \mathcal{Q}$ which occurs infinitely often in B. Since $\rhd_\mu$ is well-founded, the infinite sequence must also contain infinitely many $\hookrightarrow_{\mathcal{R},\mu}$-steps. By making repeated use of the fact that $\rhd_\mu \circ \hookrightarrow_{\mathcal{R},\mu} \subseteq \hookrightarrow_{\mathcal{R},\mu} \circ \rhd_\mu$, we obtain an infinite $\hookrightarrow_{\mathcal{R},\mu}$-sequence starting from $\pi(t_0)$. Thus, $\pi(t_0)$ is not μ-terminating with respect to $\mathcal{R}$. Since $\pi(f) \in \mu(f)$ and hence $t_0 \rhd_\mu \pi(t_0)$, this implies that t_0 is not μ-terminating (use Lemma 1(1)). This contradicts μ-termination of t_0.

Hence, $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_{\pi,\rhd}$ and B is an infinite minimal $(\mathcal{P} - \mathcal{P}_{\pi,\rhd_\mu}, \mathcal{R}, \mu)$-chain. This contradicts our initial argument. □

Example 22 (Proof of termination of the main example)**.** Consider the termination problems obtained in Example 15 for the CS-TRS in Example 1:

$$\tau_1 = (\{(1)\}, \mathcal{R}, \mu^\sharp), \quad \tau_2 = (\{(17)\}, \mathcal{R}, \mu^\sharp), \quad \tau_3 = (\{(21)\}, \mathcal{R}, \mu^\sharp), \quad \text{and} \quad \tau_4 = (\{(23)\}, \mathcal{R}, \mu^\sharp)$$

We apply $\mathsf{Proc}_{\mathit{subNColl}}$ to all such problems. For τ_1, with $\pi(\mathtt{ADD}) = 1$, we have $\pi(\mathtt{ADD}(\mathtt{s}(n), m)) = \mathtt{s}(n) \rhd_\mu n = \pi(\mathtt{ADD}(n, m))$. Now, $\mathsf{Proc}_{\mathit{subNColl}}(\tau_1) = \{(\varnothing, \mathcal{R}, \mu^\sharp)\}$. With $\mathsf{Proc}_{\mathit{Fin}}$ we conclude that τ_1 is finite. Since this can be done for τ_2, τ_3, and τ_4, the μ-termination of $\mathcal{R}$ is proved.

The following examples show that if $\mathcal{P}$ contains collapsing rules, then Theorem 12 does not hold.

Example 23. Consider the two TRSs

$$\mathcal{R} : \mathtt{h}(x) \to \mathtt{f}(\mathtt{g}(\mathtt{h}(x))) \quad \textit{and} \quad \mathcal{P} : \mathtt{f}(\mathtt{g}(x)) \to x$$

Let μ be given by $\mu(f) = \{1, \ldots, k\}$ for all symbols f. Note that, $\mathit{Root}(\mathcal{P}) = \{\mathtt{f}\}$ and $\mathcal{D} = \{\mathtt{h}\}$ are disjoint. By using the projection $\pi(\mathtt{f}) = 1$, we get $\pi(\mathtt{f}(\mathtt{g}(x))) = \mathtt{g}(x) \rhd_\mu x$. After removing the pair in $\mathcal{P}$, a finite CS problem $(\varnothing, \mathcal{R}, \mu)$ is obtained. However, $(\mathcal{P}, \mathcal{R}, \mu)$ is *not* finite:

$$\underline{\mathtt{f}(\mathtt{g}(\mathtt{h}(x)))} \hookrightarrow_{\mathcal{P},\mu} \underline{\mathtt{h}(x)} \hookrightarrow_{\mathcal{R},\mu} \underline{\mathtt{f}(\mathtt{g}(\mathtt{h}(x)))} \hookrightarrow_{\mathcal{P},\mu} \cdots$$

In the following theorem, we show how to use the subterm criterion to *remove* all collapsing pairs from $\mathcal{P}$.

Theorem 13 (Subterm processor for collapsing pairs)**.** *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs such that $\mathcal{P}_\mathcal{G}$ contains no collapsing rule, $\mathit{Root}(\mathcal{P}) \cap \mathcal{D} = \varnothing$, and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let π be a simple projection for $\mathcal{P}$ such that*

1. *$\pi(\mathcal{P}) \subseteq \trianglerighteq_\mu$, and*
2. *whenever $\mathcal{P}_\mathcal{X} \neq \varnothing$, we have $\pi(f^\sharp) \in \mu(f^\sharp) \cap \mu(f)$ for all $f \in \mathcal{H}_\mathcal{P}$.*

Then, the processor $\mathsf{Proc}_{\mathit{subColl}}$ given by

$$\mathsf{Proc}_{\mathit{subColl}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)\} & \textit{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \textit{otherwise} \end{cases}$$

is sound and complete.

Proof. Completeness is obvious because $\mathcal{P}_\mathcal{G} \subseteq \mathcal{P}$. For soundness, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A but there is no infinite minimal $(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)$-chain. Since $\mathcal{P}$ is finite, we can assume that

there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B which is an infinite minimal $(\mathcal{Q}, \mathcal{R}, \mu)$-chain where all pairs in $\mathcal{Q}$ are infinitely often used and $\mathcal{Q}$ contains some collapsing pair $u \to x \in \mathcal{Q}_{\mathcal{X}}$. Assume that B is

$$t_0 \hookrightarrow^*_{\mathcal{R},\mu} s_1 \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_1 \hookrightarrow^*_{\mathcal{R},\mu} s_2 \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^\sharp_\mu t_2 \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

where there is a substitution σ such that, for all $i \geqslant 1$, $s_i = \sigma(u_i)$ for some $u_i \to v_i \in \mathcal{P}$, and

1. if $v_i \notin \mathcal{X}$, then $t_i = \sigma(v_i)$,
2. if $v_i = x_i \in \mathcal{X}$, then $x_i \notin \mathcal{V}ar^\mu(u_i)$ and $t_i = r_i^\sharp$ for some $r_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that $\sigma(x_i) = C_i[r_i]_{p_i}$ for some $C_i[\,]_{p_i}$ and $p_i \in \mathcal{P}os^\mu(C_i[\,]_{p_i})$ such that $\mathrm{sprefix}_{C_i[\,]_{p_i}}(p_i) \subseteq \mathcal{F}$ and $r_i = \theta_i(\bar{r}_i)$ for some $\bar{r}_i \in \mathcal{NHT}_{\mathcal{Q}}$ and substitution θ_i.

Since we can freely choose the starting term of B, w.l.o.g. we assume that t_0 is a particular case of the second alternative above, i.e., there is a collapsing pair $u_0 \to x_0$ such that $\sigma(x_0) \trianglerighteq_\mu r_0$ and $t_0 = r_0^\sharp$. Note that, for all $i \geqslant 1$, $root(s_i) \in Root(\mathcal{P})$ because $root(u_i) \in Root(\mathcal{P})$. Furthermore, for all $i \geqslant 0$, $root(t_i) \in Root(\mathcal{P})$ because:

1. If $u_i \to v_i \in \mathcal{Q}_{\mathcal{G}}$, then $root(v_i) \in Root(\mathcal{P})$ and $t_i = \sigma(v_i)$.
2. If $u_i \to v_i \in \mathcal{Q}_{\mathcal{X}}$, then $root(t_i) \in \mathcal{D}^\sharp$; since $t_i \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}$ and $\mathcal{D}^\sharp \cap \mathcal{F} = \varnothing$ (see Remark 4), rewritings with $\mathcal{R}$ cannot remove the marked root symbol in t_i; hence, we can further conclude $root(t_i) = root(s_{i+1}) \in Root(\mathcal{P})$.

Therefore, we can apply π to s_{i+1} and t_i for all $i \geqslant 0$. Moreover, since $t_i \hookrightarrow^*_{\mathcal{R},\mu} s_{i+1}$ for all $i \geqslant 0$ and $Root(\mathcal{P}) \cap \mathcal{D} = \varnothing$, we can actually write $t_i \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathcal{R},\mu} s_{i+1}$. Hence, $\pi(t_i) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_{i+1})$ and also $root(t_i) = root(s_{i+1})$ for all $i \geqslant 0$.

Since $u \to x \in \mathcal{Q}_{\mathcal{X}}$ and B is infinite, it must be $\mathcal{H}_{\mathcal{Q}} \neq \varnothing$ (hence $\mathcal{H}_{\mathcal{P}} \neq \varnothing$). Thus, we have $\pi(f^\sharp) \in \mu(f)$ for all $f \in \mathcal{H}_{\mathcal{Q}} \subseteq \mathcal{H}_{\mathcal{P}}$. Then, since $root(t_i) = root(s_{i+1})$ and all pairs in $\mathcal{Q}$ occur infinitely often in B, we can assume that $root(t_0) = f$. Furthermore, since A is minimal, we can assume that t_0 is μ-terminating. We have that $\pi(u_i) \trianglerighteq_\mu \pi(v_i)$ for all $u_i \to v_i \in \mathcal{Q}$. Now we distinguish two cases:

1. If $u_i \to v_i \in \mathcal{Q}_{\mathcal{G}}$, then $s_i = \sigma(u_i)$ and $t_i = \sigma(v_i)$. By stability of $\trianglerighteq_\mu$ we have $\pi(s_i) \trianglerighteq_\mu \pi(t_i)$.
2. If $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_{\mathcal{X}}$, then $s_i = \sigma(u_i)$ and there is a term r_i, such that $\sigma(x_i) \trianglerighteq_\mu r_i$ and $r_i^\sharp = t_i$. Since $\pi(u_i) \trianglerighteq_\mu x_i$, by stability of $\trianglerighteq_\mu$ we have

   $$\pi(s_i) = \pi(\sigma(u_i)) = \sigma(\pi(u_i)) \trianglerighteq_\mu \sigma(x_i) \trianglerighteq_\mu r_i.$$

   Note that $f_i = root(r_i) = root(\bar{r}_i) \in \mathcal{H}_{\mathcal{P}}$. Since $\pi(t_{i+1}) = t_i|_{\pi(f_i^\sharp)} = r_i^\sharp|_{\pi(f_i^\sharp)} = r_i|_{\pi(f_i^\sharp)}$ and $\pi(f_i^\sharp) \in \mu(f_i)$, we have that $r_i \rhd_\mu \pi(t_i)$ and thus $\pi(s_i) \rhd_\mu \pi(t_i)$.

Therefore, by applying the simple projection π, the sequence B is transformed into an infinite $\hookrightarrow_{\mathcal{R},\mu} \cup \rhd_\mu$-sequence B'

$$\pi(t_0) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_1) \trianglerighteq_\mu \pi(t_1) \hookrightarrow^*_{\mathcal{R},\mu} \pi(s_2) \trianglerighteq_\mu \pi(t_2) \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

Since $u \to x$ occurs infinitely often in B, and by case (2) above, B' contains infinitely many $\rhd_\mu$ steps, starting from $\pi(t_0)$. Since $\rhd_\mu$ is well-founded, the infinite sequence must also contain infinitely many $\hookrightarrow_{\mathcal{R},\mu}$-steps. By making repeated use of the fact that $\rhd_\mu \circ \hookrightarrow_{\mathcal{R},\mu} \subseteq \hookrightarrow_{\mathcal{R},\mu} \circ \rhd_\mu$, we obtain an infinite $\hookrightarrow_{\mathcal{R},\mu}$-sequence starting from $\pi(t_0)$. Thus, $\pi(t_0)$ is not μ-terminating with respect to $\mathcal{R}$. Since $\pi(f^\sharp) \in \mu(f^\sharp)$ and hence $t_0 \rhd_\mu \pi(t_0)$, this implies that t_0 is not μ-terminating (use Lemma 1(1)). This contradicts μ-termination of t_0. Therefore, $\mathcal{Q}$ cannot contain collapsing pairs. This contradicts our initial assumption $u \to x \in \mathcal{Q}$. □

Remark 13. The use of Theorem 13 only makes sense if $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{G}} \cup \mathcal{P}^1_{\mathcal{X}}$. If $u \to x \in \mathcal{P}_{\mathcal{X}} - \mathcal{P}^1_{\mathcal{X}}$ for some $u = f(u_1, \ldots, u_k)$, then for all $i \in \{1, \ldots, k\}$, whenever $x \in \mathcal{V}ar(u_i)$ we have $i \in \mu(f)$ and $u_i \not\rhd_\mu x$. Thus, there is no simple projection π such that $\pi(u) \trianglerighteq_\mu x$.

Example 24. Consider the following TRS $\mathcal{R}$:

$$\mathtt{g}(x, y) \to \mathtt{f}(x, y)$$
$$\mathtt{f}(\mathtt{c}(x), y) \to \mathtt{g}(x, \mathtt{g}(y, y))$$

together with the replacement map μ given by $\mu(\mathtt{c}) = \mu(\mathtt{g}) = \{1\}$ and $\mu(\mathtt{f}) = \varnothing$. The CSDPs are:

$$\mathtt{G}(x, y) \to \mathtt{F}(x, y) \tag{33}$$

$$\mathtt{F}(\mathtt{c}(x), y) \to \mathtt{G}(x, \mathtt{g}(y, y)) \tag{34}$$

$$\mathtt{F}(\mathtt{c}(x), y) \to x \tag{35}$$

and all of them are part of the only SCC $\mathcal{P} = \{(33), (34), (35)\}$ in the CSDG of $(\mathcal{R}, \mu)$. Note that $\mathcal{NHT}_{\mathcal{P}} = \{\mathtt{g}(y, y)\}$; hence $\mathcal{H}_{\mathcal{P}} = \{\mathtt{g}\}$. Consider the simple projection π given by $\pi(\mathtt{F}) = \pi(\mathtt{G}) = 1$. Note that $\pi(\mathtt{G}) \in \mu^{\sharp}(\mathtt{G}) \cap \mu^{\sharp}(\mathtt{g})$ as required by Theorem 13. We have

- $\pi(\mathtt{G}(x, y)) = x \trianglerighteq_{\mu} x = \pi(\mathtt{F}(x, y))$
- $\pi(\mathtt{F}(\mathtt{c}(x), y)) = \mathtt{c}(x) \rhd_{\mu} x = \pi(\mathtt{G}(x, \mathtt{g}(y, y)))$,
- $\pi(\mathtt{F}(\mathtt{c}(x), y)) = \mathtt{c}(x) \rhd_{\mu} x = \pi(x)$

We use $\mathsf{Proc}_{subColl}$ to remove (35) from $\mathcal{P}$ and obtain a new problem $(\{(33), (34)\}, \mathcal{R}, \mu^{\sharp})$. Then, $\mathsf{Proc}_{subNColl}$ applies and yields $(\{(33)\}, \mathcal{R}, \mu^{\sharp})$. With Proc_{SCC}, we conclude the μ-termination of $\mathcal{R}$.

The following result provides a kind of generalization of the subterm criterion to simple projections that only take *non-μ-replacing* arguments.

Theorem 14 (Non-μ-replacing projection processor)**.** *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs such that $\mathcal{P}_{\mathcal{G}}$ contains no collapsing rule, $Root(\mathcal{P}) \cap \mathcal{D} = \emptyset$, and $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$. Let $\gtrsim$ be a stable quasi-ordering on terms whose strict and stable part $>$ is well-founded and π be a simple projection for $\mathcal{P}$ such that*

1. *for all $f \in Root(\mathcal{P})$, $\pi(f) \notin \mu(f)$,*
2. *$\pi(\mathcal{P}) \subseteq \gtrsim$,*
3. *whenever $\mathcal{NHT}_{\mathcal{P}} \neq \emptyset$ and $\mathcal{P}_{\mathcal{X}} \neq \emptyset$, we have that $\mathcal{E}mb^{\mu}(\mathcal{F}) \subseteq \gtrsim$ and $t \gtrsim t|_{\pi(root(t)^{\sharp})}$ for all $t \in \mathcal{NHT}_{\mathcal{P}}$.*

Let $\mathcal{P}_{>} = \{u \to v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$. Then, the processor Proc_{NRP} given by

$$\mathsf{Proc}_{NRP}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_{>}, \mathcal{R}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. Completeness is obvious because $\mathcal{P} - \mathcal{P}_{>} \subseteq \mathcal{P}$. For soundness, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain A but there is no infinite minimal $(\mathcal{P} - \mathcal{P}_{>}, \mathcal{R}, \mu)$-chain. Since $\mathcal{P}$ is finite, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B

$$\sigma(u_1) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^{\sharp}_{\mu} t_1 \hookrightarrow^{*}_{\mathcal{R},\mu} \sigma(u_2) \overset{\Lambda}{\hookrightarrow}_{\mathcal{Q},\mu} \circ \trianglerighteq^{\sharp}_{\mu} t_2 \hookrightarrow^{*}_{\mathcal{R},\mu} \cdots$$

for some substitution σ and pairs $u_i \to v_i \in \mathcal{Q}$, and

1. if $v_i \notin \mathcal{X}$, then $t_i = \sigma(v_i)$, and
2. if $v_i = x_i \in \mathcal{X}$, then $x_i \notin \mathcal{V}ar^{\mu}(u_i)$ and $t_i = s_i^{\sharp}$ for some s_i such that $\sigma(x_i) = C_i[s_i]_{p_i}$ for some $C_i[\ ]_{p_i}$ and $p_i \in \mathcal{P}os^{\mu}(C_i[\ ]_{p_i})$ such that $\mathrm{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ and $s_i = \theta_i(\overline{s}_i)$ for some $\overline{s}_i \in \mathcal{NHT}_{\mathcal{P}}$ and substitution θ_i.

Furthermore, *all pairs* in $\mathcal{Q}$ are used infinitely often in B. As discussed in the proof of Theorem 12, for all $i \geqslant 1$, $root(t_i) \in Root(\mathcal{P})$, $\pi(t_i) \hookrightarrow^{*}_{\mathcal{R},\mu} \pi(\sigma(u_{i+1}))$ and also $root(t_i) = root(u_{i+1})$ for all $i \geqslant 1$. No pair $u \to v \in \mathcal{Q}$ satisfies that $\pi(u) > \pi(v)$. Otherwise, by applying the simple projection π to the sequence B, we get a contradiction as follows:

1. Since $\pi(f) \notin \mu(f)$ for all $f \in Root(\mathcal{Q})$, no μ-rewritings are possible on the subterm $\pi(t_i)$ of t_i. Therefore, for all $i \geqslant 1$, $\pi(t_i) = \pi(\sigma(u_{i+1})) = \sigma(\pi(u_{i+1}))$.
2. Due to $\pi(u_i) \gtrsim \pi(v_i)$ and by stability of $\gtrsim$, we have that $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \gtrsim \sigma(\pi(v_i))$. Now, we distinguish two cases:
   (a) If $u_i \to v_i \in \mathcal{Q}_{\mathcal{G}}$, then $\pi(t_i) = \pi(\sigma(v_i)) = \sigma(\pi(v_i))$. Thus, $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$.
   (b) If $u_i \to v_i \in \mathcal{Q}_{\mathcal{X}}$, then $\sigma(\pi(v_i)) = \sigma(x_i)$. We have that $\sigma(x_i) \gtrsim s_i$ (because $\mathcal{E}mb^{\mu}(\mathcal{F}) \subseteq \gtrsim$). Let $f = root(u_{i+1}) = root(t_i) = root(\overline{s}_i^{\sharp})$. Since $t \gtrsim t|_{\pi(root(t)^{\sharp})}$ for all $t \in \mathcal{NHT}_{\mathcal{P}}$, by stability, we have $s_i = \theta_i(\overline{s}_i) \gtrsim \theta_i(\overline{s}_i|_{\pi(f)}) = \theta_i(\overline{s}_i)|_{\pi(f)} = s_i|_{\pi(f)}$. Since $s_i|_{\pi(f^{\sharp})} = t_i|_{\pi(f^{\sharp})} = \pi(t_i)$, we have $s_i \gtrsim \pi(t_i)$. Hence, $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$.

Thus, we always have $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$. We obtain an infinite $\gtrsim$ sequence

$$\pi(\sigma(u_1)) \gtrsim \pi(t_1) = \pi(\sigma(u_2)) \gtrsim \pi(t_2) \cdots$$

Since pairs in $\mathcal{Q}$ occur infinitely often, this sequence contains infinitely many $>$ steps starting from $\pi(\sigma(u_1))$. This contradicts the well-foundedness of $>$.

Therefore, $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_>$, i.e., B is an infinite minimal $(\mathcal{P} - \mathcal{P}_>, \mathcal{R}, \mu)$-chain. This contradicts our initial assumption. □

Example 25. Consider the CS-TRS $(\mathcal{R}, \mu)$ in Example 10. $\mathsf{DP}(\mathcal{R}, \mu)$ is:

$$\mathtt{G}(x) \to \mathtt{H}(x) \qquad \mathtt{H}(\mathtt{d}) \to \mathtt{G}(\mathtt{c})$$

where $\mu^\sharp(\mathtt{G}) = \mu^\sharp(\mathtt{H}) = \varnothing$. The dependency graph contains a single cycle that includes both pairs. The only simple projection is $\pi(\mathtt{G}) = \pi(\mathtt{H}) = 1$. Since $\pi(\mathtt{G}(x)) = \pi(\mathtt{H}(x))$, we only need to guarantee that $\pi(\mathtt{H}(\mathtt{d})) = \mathtt{d} > \mathtt{c} = \pi(\mathtt{G}(\mathtt{c}))$ holds for a stable and well-founded ordering $>$ (e.g., an RPO with $\mathtt{d} > \mathtt{c}$).

Theorem 15 (Non-μ-replacing projection processor II)**.** *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs such that $\mathcal{P}_\mathcal{G}$ contains no collapsing rule, $Root(\mathcal{P}) \cap \mathcal{D} = \varnothing$, and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let $\gtrsim$ be a stable quasi-ordering on terms whose strict and stable part $>$ is well-founded and let π be a simple projection for $\mathcal{P}$ such that*

1. *for all $f \in Root(\mathcal{P})$, $\pi(f) \notin \mu(f)$,*
2. *$\pi(\mathcal{P}) \subseteq \gtrsim$,*
3. *whenever $\mathcal{NHT}_\mathcal{P} \neq \varnothing$ and $\mathcal{P}_\mathcal{X} \neq \varnothing$, we have that $\mathcal{E}mb^\mu(\mathcal{F}) \subseteq \gtrsim$ and $t > t|_{\pi(root(t)^\sharp)}$ for all $t \in \mathcal{NHT}_\mathcal{P}$.*

Then, the processor Proc_{NRP2} *given by*

$$\mathsf{Proc}_{NRP2}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_\mathcal{G}, \mathcal{R}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

12. Narrowing transformation

The starting point of a proof of μ-termination of a TRS $\mathcal{R}$ is the computation of the estimated CSDG, $\mathsf{EDG}(\mathcal{R}, \mu)$, followed by the use of the SCC processor (Theorem 6). The estimation of the graph can lead to *overestimating* the arcs that connect two CSDPs.

Example 26. Consider the following example [50, Proposition 7]:

$$\mathtt{f}(\mathtt{0}) \to \mathtt{cons}(\mathtt{0}, \mathtt{f}(\mathtt{s}(\mathtt{0}))) \qquad \mathtt{p}(\mathtt{s}(x)) \to x$$
$$\mathtt{f}(\mathtt{s}(\mathtt{0})) \to \mathtt{f}(\mathtt{p}(\mathtt{s}(\mathtt{0})))$$

together with $\mu(\mathtt{f}) = \mu(\mathtt{p}) = \mu(\mathtt{s}) = \mu(\mathtt{cons}) = \{1\}$ and $\mu(\mathtt{0}) = \varnothing$. $\mathsf{DP}(\mathcal{R}, \mu)$ consists of the pairs:

$$\mathtt{F}(\mathtt{s}(\mathtt{0})) \to \mathtt{F}(\mathtt{p}(\mathtt{s}(\mathtt{0}))) \tag{36}$$
$$\mathtt{F}(\mathtt{s}(\mathtt{0})) \to \mathtt{P}(\mathtt{s}(\mathtt{0})) \tag{37}$$

The *estimated* CS-dependency graph contains one cycle: {(36)}. However, this cycle does *not* belong to the CS-dependency graph because there is no way to μ-rewrite $\mathtt{F}(\mathtt{p}(\mathtt{s}(\mathtt{0})))$ into $\mathtt{F}(\mathtt{s}(\mathtt{0}))$.

As already observed by Arts and Giesl for the standard case [10], in our case, the overestimation comes when a (noncollapsing) pair $u_i \to v_i$ is followed in a chain by a second one $u_{i+1} \to v_{i+1}$ and v_i and u_{i+1} are not directly unifiable, i.e., at least one μ-rewriting step is needed to μ-reduce $\sigma(v_i)$ to $\sigma(u_{i+1})$. Then, we always have $\sigma(v_i) \hookrightarrow_{\mathcal{R},\mu^\sharp} \sigma(v_i') \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$. Then, v_i' is a one-step μ-narrowing of v_i, and we could require $u_i \sqsupset v_i'$ (which could be easier to prove) instead of $u_i \sqsupset v_i$. Furthermore, we could discover that v_i has *no μ-narrowings*. In this case, we know that no chain starts from $\sigma(v_i)$.

We can be more precise when connecting two pairs $u \to v$ and $u' \to v'$ in a chain if we perform all the possible one-step μ-narrowings on v in order to develop the possible reductions from $\sigma(v)$ to $\sigma(u')$. Then, we obtain new terms $v_1, \ldots, v_n$, which are one-step μ-narrowings of v using unifiers θ_i (i.e., $v \rightsquigarrow_{\mathcal{R},\mu,\theta_i} v_i$) for $i \in \{1, \ldots, n\}$, respectively. These unifiers are also applied to the left-hand side u of the pair $u \to v$. Therefore, we can replace a pair $u \to v$ by all its (one-step) *μ-narrowed pairs* $\theta_1(u) \to v_1, \ldots, \theta_n(u) \to v_n$.

As in [10,35], a pair $u \to v \in \mathcal{P}$ can only be replaced by its μ-narrowings if the right-hand side v does not unify with any left-hand side u' of a (possibly renamed) pair $u' \to v' \in \mathcal{P}$ (note that this excludes pairs $u \to v$ with $v \in \mathcal{X}$). Moreover, the term v must be *linear*. We need to demand linearity instead of (the apparently more natural) μ-linearity (i.e., something like "no multiple μ-*replacing* occurrences of the same variable are allowed").

Example 27. The following TRS is used in [10] to motivate the requirement of linearity.

$$\begin{aligned} \mathtt{f}(\mathtt{s}(x)) &\to \mathtt{f}(\mathtt{g}(x,x)) \\ \mathtt{g}(\mathtt{0},\mathtt{1})) &\to \mathtt{s}(\mathtt{0}) \\ \mathtt{0} &\to \mathtt{1} \end{aligned}$$

We make it a CS-TRS by adding a replacement map μ given by $\mu(\mathtt{f}) = \mu(\mathtt{s}) = \{1\}$, and $\mu(\mathtt{g}) = \{2\}$. The only cycle in the CSDG consists of the CSDP

$$\mathtt{F}(\mathtt{s}(x)) \to \mathtt{F}(\mathtt{g}(x,x)).$$

If linearity of the right-hand sides is not required for μ-narrowing CSDPs, then this pair will be removed, since $\mathtt{F}(\mathtt{g}(x,x))$ and the (renamed version of) the left-hand side $\mathtt{F}(\mathtt{s}(x'))$ do not unify. Thus, there are no μ-narrowings. However, the system is *not* μ-*terminating*:

$$\underline{\mathtt{f}(\mathtt{s}(\mathtt{0}))} \hookrightarrow \mathtt{f}(\mathtt{g}(\mathtt{0},\underline{\mathtt{0}})) \hookrightarrow \mathtt{f}(\underline{\mathtt{g}(\mathtt{0},\mathtt{1})}) \hookrightarrow \underline{\mathtt{f}(\mathtt{s}(\mathtt{0}))} \ldots$$

The problem is that the μ-reduction from $\sigma(\mathtt{F}(\mathtt{g}(x,x)))$ to $\sigma(\mathtt{F}(\mathtt{s}(x')))$ takes place 'in σ', and, therefore, it cannot be captured by μ-narrowing. Note that $\mathtt{F}(\mathtt{g}(x,x))$ is "μ-linear".

Another restriction to take into account when μ-narrowing a noncollapsing pair $u \to v$ is that the μ-replacing variables in v have to be μ-replacing in u as well (this corresponds with the notion of conservativeness). Furthermore, they cannot be both μ-replacing and non-μ-replacing at the same time. This corresponds to the following definition.

Definition 14 (Strongly conservative [29])**.** Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. A rule $l \to r$ is strongly μ-conservative if it is μ-conservative and $\mathcal{V}ar^{\mu}(l) \cap \mathcal{V}ar^{\not\mu}(l) = \mathcal{V}ar^{\mu}(r) \cap \mathcal{V}ar^{\not\mu}(r) = \varnothing$.

The following result shows that, under these conditions, the set of CSDPs can be safely replaced by their μ-narrowings.

Theorem 16 (Narrowing processor)**.** *Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}}$. Let $u \to v \in \mathcal{P}$ be such that*

1. *v is linear,*
2. *for all $u' \to v' \in \mathcal{P}$ (with possibly renamed variables), v and u' do not unify.*

Let $\mathcal{Q} = (\mathcal{P} - \{u \to v\}) \cup \{u' \to v' \mid u' \to v' \text{ is a } \mu\text{-narrowing of } u \to v\}$. Then, the processor Proc_{narr} given by

$$\mathsf{Proc}_{narr}(\mathcal{P},\mathcal{R},\mu) = \begin{cases} \{(\mathcal{Q},\mathcal{R},\mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P},\mathcal{R},\mu)\} & \text{otherwise} \end{cases}$$

is

1. *sound whenever $u \to v$ is strongly conservative,*
2. *complete in all cases.*

Proof. The proof of this theorem is analogous to the proof of [35, Theorem 31], which we adapt here. For soundness, we prove that given a minimal $(\mathcal{P},\mathcal{R},\mu)$-chain "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$", there is a μ-narrowing v' of v with the mgu θ such that "$\ldots, u_1 \to v_1, \theta(u) \to v', u_2 \to v_2, \ldots$" is also a minimal $(\mathcal{Q},\mathcal{R},\mu)$-chain. Hence, every infinite minimal $(\mathcal{P},\mathcal{R},\mu)$-chain yields an infinite minimal $(\mathcal{Q},\mathcal{R},\mu)$-chain.

If "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$" is a minimal $(\mathcal{P},\mathcal{R},\mu)$-chain, then there is a substitution σ such that for all pairs $s \to t$ in the chain,

1. if $s \to t \in \mathcal{P}_{\mathcal{G}}$, then $\sigma(t)$ is μ-terminating and it μ-reduces to the instantiated left-hand side $\sigma(s')$ of the next pair $s' \to t'$ in the chain
2. if $s \to t = s \to x \in \mathcal{P}_{\mathcal{X}}$ then, $\sigma(x)$ has a μ-replacing subterm s_0, $\sigma(x) \trianglerighteq_{\mu} s_0$ such that $s_0^{\sharp}$ is μ-terminating and it μ-reduces to the instantiated left-hand side $\sigma(s')$ of the next pair $s' \to t'$ in the chain; furthermore, there is $\bar{s}_0 \in \mathcal{NHT}(\mathcal{R},\mu)$ such that $s_0 = \theta_0(\bar{s}_0)$ for some substitution θ_0.

Assume that σ is a substitution satisfying the above requirements and such that the length of the sequence $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)$ is *minimum*. Note that the length of this μ-reduction sequence cannot be zero because v and u_2 do not unify, that is, $\sigma(v) \neq \sigma(u_2)$. Hence, there is a term q such that $\sigma(v) \hookrightarrow_{\mathcal{R},\mu} q \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)$. We consider two possible cases:

1. The reduction $\sigma(v) \hookrightarrow_{\mathcal{R},\mu} q$ takes place within a binding of σ, i.e., there is a term r, a μ-replacing variable position $p \in \mathcal{P}os^{\mu}_{\mathcal{X}}(v)$, and a μ-replacing variable $x \in \mathcal{V}ar^{\mu}(v)$ such that $v|_p = x$, $q = \sigma(v[r]_p)$ and $\sigma(x) \hookrightarrow_{\mathcal{R},\mu} r$. Since v is linear, x occurs only once in v. Thus, $q = \sigma'(v)$ for the substitution σ' with $\sigma'(x) = r$ and $\sigma'(y) = \sigma(y)$ for all variables $y \neq x$. As we assume that all occurrences of pairs in the chain are variable disjoint, $\sigma'(x)$ behaves like σ for all pairs except $u \to v$. We have $\sigma(z) \hookrightarrow^*_{\mathcal{R},\mu} \sigma'(z)$ for all $z \in \mathcal{X}$. Since $u \to v$ is strongly conservative, we also have $\sigma(u) \hookrightarrow^*_{\mathcal{R},\mu} \sigma'(u)$ because all occurrences of x in u must be μ-replacing. Hence, if $u_1 \to v_1 \in \mathcal{P}_{\mathcal{G}}$ we have

   $$\sigma'(v_1) = \sigma(v_1) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) \hookrightarrow^*_{\mathcal{R},\mu} \sigma'(u)$$

   and if $u_1 \to v_1 \in \mathcal{P}_{\mathcal{X}}$, then there is $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that

   $$\sigma'(v_1) = \sigma(v_1) \trianglerighteq_{\mu} s_1 \text{ and } s_1^{\sharp} \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) \hookrightarrow^*_{\mathcal{R},\mu} \sigma'(u)$$

   and, in both cases,

   $$\sigma'(v) = q \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2) = \sigma'(u_2).$$

   Note that, by minimality and because $u \to v \in \mathcal{P}_{\mathcal{G}}$, $\sigma(v)$ is $(\mathcal{R}, \mu)$-terminating and, since $\sigma(v) \hookrightarrow_{\mathcal{R},\mu} q$, the term q is $(\mathcal{R}, \mu)$-terminating as well. Therefore, $\sigma'(x) = q$ is $(\mathcal{R}, \mu)$-terminating and σ' satisfies the two conditions above. Since the length of the sequence $\sigma'(v) \hookrightarrow^*_{\mathcal{R},\mu} \sigma'(u_2)$ is shorter than the sequence $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)$, we obtain a contradiction and we conclude that the μ-reduction $\sigma(v) \hookrightarrow_{\mathcal{R},\mu} q$ cannot take place in a binding of σ.
2. The reduction $\sigma(v) \hookrightarrow_{\mathcal{R},\mu} q$ 'touches' v, i.e., there is a nonvariable position $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(v)$, and a rewrite rule $l \to r \in \mathcal{R}$ such that $\sigma(v|_p) = \rho(l)$, for some substitution ρ and

   $$\sigma(v) = \sigma(v)[\sigma(v|_p)]_p = \sigma(v)[\rho(l)]_p \hookrightarrow_{\mathcal{R},\mu} \sigma(v)[\rho(r)]_p = q$$

   Since we can assume that variables in l are fresh, we can extend σ to behave like ρ on variables in l. Thus, $\sigma(l) = \sigma(v|_p)$, i.e, l and $v|_p$ unify and there is a mgu θ and a substitution τ satisfying $\sigma(x) = \tau(\theta(x))$ for all variables x. We have that v μ-narrows to $\theta(v)[\theta(r)]_p = v'$ with unifier θ. Again, we can extend σ to behave like τ on the variables of $\theta(u)$ and v'. Therefore, if $u_1 \to v_1 \in \mathcal{P}_{\mathcal{G}}$ we have

   $$\sigma(v_1) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) = \tau(\theta(u)) = \sigma(\theta(u))$$

   and if $u_1 \to v_1 \in \mathcal{P}_{\mathcal{X}}$, then there is $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ such that

   $$\sigma(v_1) = \sigma(x) \trianglerighteq_{\mu} s_1 \text{ and } s_1^{\sharp} \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) = \tau(\theta(u)) = \sigma(\theta(u))$$

   and

   $$\sigma(v') = \tau(v') = \tau(\theta(v))[\tau(\theta(r))]_p = \sigma(v)[\sigma(r)]_p = \sigma(v)[\rho(r)]_p = q \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)$$

   Hence, "$\ldots, u_1 \to v_1, \theta(u) \to v', u_2 \to v_2, \ldots$" is also a minimal chain.

Completeness is also analogous to the 'completeness' part of [35, Theorem 31]. If $(\mathcal{Q}, \mathcal{R}, \mu)$ is infinite and $\mathcal{R}$ is non-μ-terminating, then $(\mathcal{P}, \mathcal{R}, \mu)$ is infinite as well. If $\mathcal{R}$ is μ-terminating, then let "$\ldots, u_1 \to v_1, \theta(u) \to v', u_2 \to v_2, \ldots$" be an infinite minimal $(\mathcal{Q}, \mathcal{R}, \mu)$-chain where v' is a one-step μ-narrowing of v using the mgu θ. We prove that "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$" is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mu)$-chain. There is a substitution σ such that

$$\sigma(v_1) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(\theta(u)) \text{ if } u_1 \to v_1 \in \mathcal{P}_{\mathcal{G}}, \text{ and}$$

$$\sigma(v_1) = \sigma(x) \trianglerighteq_{\mu} s_1 \text{ and } s_1^{\sharp} \hookrightarrow^*_{\mathcal{R},\mu} \sigma(\theta(u)) \text{ if } u_1 \to v_1 \in \mathcal{P}_{\mathcal{X}}$$

Finally, we also have

$$\sigma(v') \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2).$$

Since the variables in the pairs are pairwise disjoint, we may extend σ to behave like $\sigma(\theta(x))$ on $x \in \mathcal{V}ar(u)$ then $\sigma(u) = \sigma(\theta(u))$ and therefore

$$\sigma(v_1) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) \qquad \text{if } u_1 \to v_1 \in \mathcal{P}_\mathcal{G}, \text{ and}$$

$$\sigma(v_1) \trianglerighteq_\mu s_1 \text{ and } s_1^\sharp \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u) \text{ if } u_1 \to v_1 \in \mathcal{P}_\mathcal{X}$$

Moreover, by definition of μ-narrowing, we have $\theta(v) \hookrightarrow_{\mathcal{R},\mu} v'$. This implies that $\sigma(\theta(v)) \hookrightarrow_{\mathcal{R},\mu} \sigma(v')$, and since $\sigma(v) = \sigma(\theta(v))$, we obtain

$$\sigma(v) \hookrightarrow_{\mathcal{R},\mu} \sigma(v') \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2).$$

Since $\mathcal{R}$ is μ-terminating, $\sigma(v)$ is $(\mathcal{R}, \mu)$-terminating. Hence, "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$" is a minimal infinite $(\mathcal{P}, \mathcal{R}, \mu)$-chain as well. □

Example 28. Since the right-hand side of pair (36) in Example 26 does not unify with any (renamed) left-hand side of a CSDP (including itself) and it can be μ-narrowed at position 1 (notice that $\mu(\mathtt{f})$={1}) by using the rule $\mathtt{p}(\mathtt{s}(x)) \to x$, we can replace it by its μ-narrowed pair:

$$\mathtt{F}(\mathtt{s}(\mathtt{0})) \to \mathtt{F}(\mathtt{0}) \tag{38}$$

Now, $\mathsf{Proc}_{SCC}(\{(38)\}, \mathcal{R}, \mu) = \varnothing$ and the μ-termination of $\mathcal{R}$ is proved.

The following example shows that strong conservativeness cannot be dropped for the pair $u \to v$ to be μ-narrowed. This requirement was not taken into account in [4, Theorem 5.3].

Example 29. Consider the following[8] TRS $\mathcal{R}$:

$$\begin{aligned} \mathtt{c}(\mathtt{e}(x)) &\to \mathtt{d}(x, x) \\ \mathtt{a} &\to \mathtt{e}(\mathtt{a}) \end{aligned}$$

and $\mathcal{P}$ consisting of the following pair:

$$\mathtt{F}(\mathtt{d}(x, x)) \to \mathtt{F}(\mathtt{c}(x))$$

together with $\mu(\mathtt{c}) = \mu(\mathtt{d}) = \mu(\mathtt{F}) = \{1\}$ and $\mu(\mathtt{e}) = \varnothing$. There is an infinite $(\mathcal{P}, \mathcal{R}, \mu)$-chain:

$$\mathtt{F}(\mathtt{c}(\underline{\mathtt{a}})) \hookrightarrow_{\mathcal{R},\mu} \mathtt{F}(\underline{\mathtt{c}(\mathtt{e}(\mathtt{a}))}) \hookrightarrow_{\mathcal{R},\mu} \underline{\mathtt{F}(\mathtt{d}(\mathtt{a}, \mathtt{a}))} \hookrightarrow_{\mathcal{P},\mu} \mathtt{F}(\mathtt{c}(\underline{\mathtt{a}})) \hookrightarrow_{\mathcal{R},\mu} \cdots$$

Since $\mathtt{F}(\mathtt{c}(x))$ does not unify with any left-hand side of another pair, we can μ-narrow the pair in $\mathcal{P}$. We obtain $\mathcal{P}'$ consisting of the μ-narrowed pair

$$\mathtt{F}(\mathtt{d}(\mathtt{e}(x), \mathtt{e}(x))) \to \mathtt{F}(\mathtt{d}(x, x))$$

No infinite $(\mathcal{P}', \mathcal{R}, \mu)$-chain is possible now. Note that $\mathcal{P}$ is μ-conservative, but it is *not* strongly μ-conservative (the variable x is both μ-replacing and non-μ-replacing in $\mathtt{F}(\mathtt{d}(x, x))$).

Remark 14 (Implementing the narrowing processor)**.** In our current implementation, we apply the narrowing processor only if, after computing the (one-step) μ-narrowings of the right-hand side v of a pair $u \to v \in \mathcal{P}$, the new CS-dependency graph does not increase the number of arcs. More sophisticated strategies like (the corresponding adaptations of) the *safe transformations* in [35, Definition 33] could be considered in the future.

13. Experiments

The processors described in the previous sections were implemented as part of the tool MU-TERM. We tested the CSDP-framework in practice on the 90 examples in the Context-Sensitive Rewriting subcategory of the 2007 International Termination Competition:

http : //www.lri.fr/∼marche/termination − competition/2007/.

These 90 examples are part of the Termination Problem Data Base (TPDB, version 4.0):

http : //www.lri.fr/∼marche/tpdb/.

[8] We thank Fabian Emmes for providing this example.

Table 1
Comparison among *CSR* termination techniques.

Tool version	Proved	Total time	Average time
CSDPs	65/90	0.31 s	0.00 s
CSRPO	37/90	0.21 s	0.00 s
Polynomial orderings	27/90	0.06 s	0.00 s
Transformations	56/90	5.59 s	0.10 s

We addressed this task in three different ways:

1. We compared CSDPs with previously existing techniques for proving termination of *CSR*.
2. We compared the improvements introduced by the different CS processors which have been defined in this paper.
3. We participated in the *CSR* subcategory of the 2007 International Termination Competition.

In the following subsections, we provide more details about this experimental evaluation.

13.1. CSDPs vs. other techniques for proving termination of CSR

Several methods have been developed to prove termination of *CSR* for a given CS-TRS $(\mathcal{R}, \mu)$. Two main approaches have been investigated so far:

1. *Direct proofs*, which are based on using μ-reduction orderings (see [63]) such as the (context-sensitive) recursive path orderings [12] and polynomial orderings [26,48,49]. These are orderings $>$ on terms that can be used to directly compare the left- and right-hand sides of the rules in order to conclude the μ-termination of the TRS.
2. *Indirect proofs*, which obtain a proof of the μ-termination of $\mathcal{R}$ as a proof of termination of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation). If we are able to prove *termination* of $\mathcal{R}^{\mu}_{\Theta}$ (using the standard methods), then the μ-termination of $\mathcal{R}$ is ensured.

We used MU-TERM to compare all these techniques with respect to the aforementioned benchmark examples. The results of this comparison are summarized in Table 1.

Remark 15. A number of transformations Θ from TRSs $\mathcal{R}$ and replacement maps μ that produce TRSs $\mathcal{R}^{\mu}_{\Theta}$ have been investigated by Lucas (transformation L [43]), Zantema (transformation Z [63]), Ferreira and Ribeiro (transformation FR [22]), and Giesl and Middeldorp (transformations[9] GM, sGM, and C [30,31]), see [31,50] for recent surveys about these transformations which also include a thorough analysis about their relative power. All these transformations were considered in our experiments, so the item *"Transformations"* in Table 1 concentrates the *joint* impact of all of them.

From the benchmarks summarized in Table 1, we clearly conclude that the CSDP-framework is the most powerful technique for proving termination of *CSR*. Actually, all the examples that were solved by using CSRPO or polynomial orderings were also solved using CSDPs. With regard to transformations, there is only one example (namely, `Ex9_Luc06`, which can be solved by using transformation GM) that could not be solved with our current implementation.

Example 30. The following nonterminating TRS $\mathcal{R}$ can be used to compute the list of prime numbers by using the well-known Erathostenes sieve[10] [30]:

$$\begin{aligned}
\mathtt{primes} &\to \mathtt{sieve}(\mathtt{from}(\mathtt{s}(\mathtt{s}(\mathtt{0}))))\\
\mathtt{from}(x) &\to \mathtt{cons}(x, \mathtt{from}(\mathtt{s}(x)))\\
\mathtt{head}(\mathtt{cons}(x, y)) &\to x\\
\mathtt{sieve}(\mathtt{cons}(x, y)) &\to \mathtt{cons}(x, \mathtt{filt}(x, \mathtt{sieve}(y)))\\
\mathtt{tail}(\mathtt{cons}(x, y)) &\to y\\
\mathtt{if}(\mathtt{true}, x, y) &\to x\\
\mathtt{if}(\mathtt{false}, x, y) &\to y\\
\mathtt{filt}(\mathtt{s}(\mathtt{s}(x)), \mathtt{cons}(y, z)) &\to \mathtt{if}(\mathtt{div}(\mathtt{s}(\mathtt{s}(x)), y), \mathtt{filt}(\mathtt{s}(\mathtt{s}(x)), z), \mathtt{cons}(y, \mathtt{filt}(\mathtt{s}(\mathtt{s}(x)), z)))
\end{aligned}$$

[9] The labels for these transformations correspond to the ones introduced in [50].
[10] Without appropriate rules for defining symbol `div`, the TRS has no complete computational meaning. However, we take it here as given in [30] for the purpose of comparing different techniques for proving termination of *CSR* by transformation.

Table 2
Comparison among CS processors.

Tool version	Narrowing	Non-μ-replacing projection	Subterm	Proved	Total time	Average time
1	No	No	No	54/90	3.00 s	0.05 s
2	No	No	Yes	62/90	0.55 s	0.01 s
3	No	Yes	No	57/90	0.82 s	0.01 s
4	No	Yes	Yes	65/90	0.49 s	0.01 s
5	Yes	No	No	54/90	3.22 s	0.06 s
6	Yes	No	Yes	62/90	2.64 s	0.04 s
7	Yes	Yes	No	57/90	1.27 s	0.02 s
8	Yes	Yes	Yes	65/90	0.31 s	0.00 s

Consider the replacement map μ for the signature $\mathcal{F}$ given by:

$$\mu(\mathtt{cons}) = \mu(\mathtt{if}) = \{1\} \text{ and } \mu(f) = \{1, \ldots, ar(f)\} \text{ for all } f \in \mathcal{F} - \{\mathtt{cons}, \mathtt{if}\}.$$

From the termination point of view, this example is interesting because, since its introduction in Giesl and Middeldorp's paper [30], no automatic proof of termination has been reported. In sharp contrast, termination of *CSR* for this TRS and replacement map μ is easily proved by using the techniques developed in this paper. In particular, the context-sensitive dependency graph contains no cycle.

13.2. Contribution of the different CS processors

In our implementation of the CSDP-framework, besides processor Proc_{SCC}, the subterm processors in Section 11 and the μ-reduction-pair CS processors in Section 10 are the most frequently used (in this order). The impact of the CS processors in Sections 11 and 12 is summarized in Table 2. Our benchmarks show that the CS processors described in Section 11 play an important role in our proofs. The subterm processors $\mathsf{Proc}_{subNColl}$ and $\mathsf{Proc}_{subColl}$ are quite efficient, but the ones that are based on simple projections for non-μ-replacing arguments (Proc_{NRP} and Proc_{NRP2}) also increase the power and the speed of the CSDP technique. Furthermore, these two groups of CS processors are complementary: the extra problems that are specifically solved by them are different. Narrowing is useful for simplifying the graph, but it does not play an important role in the benchmarks because it is only applied to solve two examples (which can be solved without narrowing as well). Furthermore, it must be used carefully because recomputing the graph can be expensive in that case. Complete details of our experiments can be found here:

http : *//zenon.dsic.upv.es/muterm/benchmarks/csdp/*.

13.3. CSDPs at the 2007 International Termination Competition

In 2007, AProVE [32] was the only tool (besides MU-TERM) implementing specific methods for proving termination of *CSR*. Both AProVE and MU-TERM participated in the *CSR* subcategory of the 2007 International Termination Competition. AProVE participated with a termination expert for *CSR* which, given a CS-TRS $(\mathcal{R}, \mu)$, successively tries different transformations Θ for proving termination of *CSR* (which are enumerated in Remark 15, i.e., $\Theta \in \{C, FR, GM, L, sGM, Z\}$). It then uses a huge variety of different and complementary techniques to prove termination of rewriting (according to the DP-framework) on the obtained TRS $\mathcal{R}^{\mu}_{\Theta}$. Actually, AProVE is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques.

However, MU-TERM's implementation of CSDPs was able to beat AProVE in the *CSR* category (MU-TERM was able to prove 68 of the 90 examples; AProVE proved 64), thus demonstrating that CSDPs are actually a very powerful technique for proving termination of *CSR*.

14. Related work

The first presentation of the context-sensitive dependency pairs was given in [3]. This paper is an extended and revised version of [3,4]. We provide complete proofs for all results,[11] and also present many examples about the use of the theory. The main conceptual differences between [3,4] and this paper are the following:

1. In this paper, we have investigated two different notions of minimal non-μ-terminating terms: the so-called *strongly minimal terms* ($\mathcal{T}_{\infty,\mu}$, which are introduced in this paper) and the *minimal terms* ($\mathcal{M}_{\infty,\mu}$), which were introduced in [3] and further investigated in [4]. The combined use of these notions leads to a better development of the theory. This has brought new essential results, remarkably Theorem 1, which is the basis (at the level of pure context-sensitive rewriting) of the new notions of CSDP and minimal chain.

[11] We report and fix some bugs in previous papers.

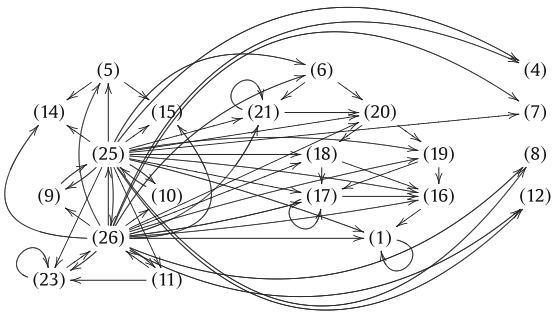


Fig. 6. Context-sensitive dependency graph of Example 1 following [3].

2. Although most of the ideas in this first part of the paper (Section 3) were present in [4, Section 3], we make some aspects explicit that were only implicit there. For instance, the essential notion of *hidden term* (a consequence of Lemma 5 which is further developed in Lemma 6 and Proposition 4) was implicit in [4, Section 3], but only the notion of *hidden symbol* was made explicit. Actually, the proofs of the aforementioned results in this paper correspond (with minor changes) to those of Lemmas 3.4 and 3.5, and Proposition 3.6 in [4], respectively.
3. The notion of context-sensitive dependency pairs was first introduced in [3, Definition 1], but the narrowing condition that we have now included for the noncollapsing CSDPs is new. This condition is inspired in the recent extension of the DP-method to Order-Sorted TRSs [53]. In this paper, we have elaborated it in depth to show that it is actually a natural requirement (see Section 3.4). In [53], it has already been shown that including 'narrowability' in the usual definition of dependency pair can be useful to automatically prove termination of rewriting. Similar considerations are valid for CSR.
4. In [3], a notion of minimal chain was introduced but not really used in the main results. Actually, the notion of minimal chain in this paper is completely different from the old one and is a consequence of the analysis of infinite μ-rewrite sequences developed in Section 3. Furthermore, in this paper, the notion of minimal chain of pairs is essential for the definition of the context-sensitive dependency graph and the development of the CSDP-framework in Section 7.
5. The notion of context-sensitive dependency graph was first introduced in [3] and further refined in [4] thanks to the introduction of the hidden symbols. The definition in this paper introduces a new refinement through the notion of 'narrowable hidden term' and shows a nice symmetry between the arcs associated to noncollapsing and collapsing pairs. Furthermore, the new definition leads to a great simplification of the computed graph: for the CS-TRS in Example 1, compare the graph in Fig. 6 (corresponding to [3]) with the new graph in Fig. 5.
6. The estimation of the CSDG in [3,4] was an adaptation of the one by Arts and Giesl [10] to the context-sensitive setting. In this paper, we have defined a new estimation of the CSDG on the basis of the most recent proposal by Giesl et al. [34].
7. The definition of a CSDP-framework for the mechanization of proofs of termination of *CSR* using CSDPs is new. A number of processors introduced here had a kind of counterpart in [3] (for instance, the use of μ-reduction orderings was formalized in [3, Theorem 4] and the subterm criterion for noncollapsing pairs was formalized in [3, Theorem 5]) or in [4] (for instance, the narrowing transformation in [4, Theorem 5.3]), but they were not formulated as processors.
8. This paper introduces a number of new processors that can be used for proving termination of *CSR*: the SCC processor, [12] the processors for filtering or transforming collapsing pairs (see Section 9), the use of argument filterings, [13] the use of the subterm criterion with collapsing pairs (Theorem 13), etc.
9. Finally, for the first time, we have considered how to *disprove* termination of *CSR* within the CSDP framework (processor Proc_{Inf} in Theorem 5).

14.1. CSDPs vs. DPs and a piece of history

The first attempt to develop a theory of dependency pairs for *CSR* started more than 10 years ago when the third author of this paper asked Thomas Arts (who was preparing the first presentation of the dependency pair method [9]) about the possibility of extending the dependency pair approach to *CSR*. Arts immediately noticed that the main problem of extending the existing results for ordinary rewriting to *CSR* was the possibility of having variables that are not replacing in the left-hand sides of the rules but that become replacing in the corresponding right-hand side. This is what we now call *migrating* variables. After this first failed attempt, the focus moved to transformations of CS-TRSs $(\mathcal{R}, \mu)$ into ordinary TRSs $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation) in such a way that termination of $\mathcal{R}^{\mu}_{\Theta}$ implies the μ-termination of $\mathcal{R}$ [31,50].

[12] This is mentioned in [3, Section 4.2] but without any formal description.
[13] This was briefly mentioned at the end of [3, Section 4.2] but was never formalized.

During the spring of 2006, MU-TERM was being revised in preparation for its participation in the 2006 International Termination Competition, which was organized by Claude Marché. The idea of adapting DPs to *CSR* came up again. A first correct version of context-sensitive dependency pairs that did *not* at the time consider collapsing pairs was the following:

Definition 15 (First preliminary version of CSDPs). Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let

$$\begin{aligned}\mathsf{DP}_1(\mathcal{R}, \mu) = &\left\{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s\right\}\\ \cup &\left\{l^\sharp \to \mathtt{MUSUBTERM}(x) \mid l \to r \in R, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\right\}\\ \cup &\{\mathtt{MUSUBTERM}(f(x_1, \ldots, x_k)) \to \mathtt{MUSUBTERM}(x_i) \mid f \in \mathcal{F}, i \in \mu(f)\}\\ \cup &\{\mathtt{MUSUBTERM}(f(x_1, \ldots, x_k)) \to f^\sharp(x_1, \ldots, x_k) \mid f \in \mathcal{D}\}\end{aligned}$$

with $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$, and $\mu^\sharp(\mathtt{MUSUBTERM}) = \varnothing$.

We handle migrating variables x by enclosing them inside a term $\mathtt{MUSUBTERM}(x)$ which (after instantiating x by means of a substitution σ) would be able to start the search for a μ-replacing subterm $s = f(s_1, \ldots, s_k)$ which (after marking its root symbol f as $f^\sharp$) is able to connect with the left-hand side of the next CSDP in a sequence. The notion of *chain* of CSDPs that was used here was essentially the standard one. All pairs were treated in the very same way and the only difference was that pairs were connected by using *CSR* instead of ordinary rewriting.

The implementation of the CSDPs in Definition 15 did not work very well in practice. The structure of pairs which dealt with migrating variables introduced many arcs in the corresponding graph and, therefore, many cycles. Thus, the following proposal was considered instead.

Definition 16 (Second preliminary version of CSDPs). Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $\mathsf{DP}_2(\mathcal{R}, \mu) = \mathsf{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu) \cup \mathsf{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu)$ where:

$$\begin{aligned}\mathsf{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu) &= \left\{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s\right\}\\ \mathsf{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu) &= \left\{l^\sharp \to U_{l,f,x}(x) \mid l \to r \in R, f \in \mathcal{D}, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\right\}\\ &\quad \cup \{U_{l,f,x}(f(x_1, \ldots, x_k)) \to f^\sharp(x_1, \ldots, x_k) \mid l \to r \in R, f \in \mathcal{D}, x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)\}\end{aligned}$$

and $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$, and $\mu^\sharp(U_{l,f,x}) = \{1\}$ for all rules $l \to r$, symbols f, and variables x originating one of these symbols.

Here, migrating variables x are enclosed inside a term $U_{l,f,x}(x)$ which (after instantiating x by means of a substitution σ) would be able to connect any μ-replacing subterm $s = f(s_1, \ldots, s_k)$ (with f a defined symbol) with the left-hand side of the next CSDP in a sequence. Note that no explicit μ-replacing subterm search is possible with this new definition of CSDP. Instead, this requirement *was moved to the definition of chain*. Now, although these dependency pairs still remain as the 'traditional ones', *a clear distinction was made between two kinds of CSDPs*: those that were obtained from the nonvariable parts of the right-hand sides of the rules ($\mathsf{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu)$ in Definition 16) and those that were introduced to treat the migrating variables ($\mathsf{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu)$ in Definition 16). Both kinds of CSDPs were clearly distinguished in the new definition of chain and the μ-subterm requirement was used to describe how chains of such CSDPs are built.

A version of MU-TERM that implemented the CSDPs in Definition 16 was submitted for participation in the *Context-Sensitive* (sub)category of the 2006 International Termination Competition (June 2006). We are grateful to Claude Marché for providing a copy of the folder where the MU-TERM outcome was stored. It is now available at the following URL:

http : *//zenon.dsic.upv.es/muterm/benchmarks/ic*10*/muterm* − 2006*/benchmarks.html*

A further evolution led to the definition of CSDP which was finally published in [3]. In sharp contrast to the standard dependency pair approach, where all dependency pairs have tuple symbols $f^\sharp$ both in the left- and right-hand sides, we finally took the definitive step to also consider *collapsing* pairs having a single *variable* in the right-hand side as the most elegant, concise and expressive way to reflect the effect of the *migrating* variables in the termination behavior of *CSR*. This is one of the most important and original contributions of the paper.

15. CSDPs vs. noncollapsing CSDPs

In [1], a transformation of collapsing pairs into 'ordinary' (i.e., noncollapsing) pairs is introduced. The transformation uses the following notion.

Definition 17 (Hiding context [1, Definition 7]). Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. The function symbol f *hides the argument* i if there is a rule $l \to r \in \mathcal{R}$ with $r \rhd_\mu f(r_1, \ldots, r_i, \ldots, r_n)$, $i \in \mu(f)$, and r_i contains a defined symbol or a variable at an active position. A context C is *hiding* iff $C = \Box$ or C has the form $f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_n)$ where f hides the argument i and C' is a hiding context.

The notion of *CSDPs* that is given in [1] is the following:

Definition 18 [1, Definition 9]. Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. If $\mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \neq \varnothing$, we introduce a fresh *unhiding tuple symbol* $\mathtt{U}$ and the following *unhiding DPs*:

- $s \to \mathtt{U}(x)$ for every $s \to x \in \mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$,
- $\mathtt{U}(f(x_1, \ldots, x_i, \ldots, x_n)) \to \mathtt{U}(x_i)$ for every function symbol f of any arity n and every $1 \leq i \leq n$ where f hides position i,
- $\mathtt{U}(t) \to t^\sharp$ for every hidden term t.

Let $\mathsf{DP}_u(\mathcal{R}, \mu)$ be the set of all unhiding DPs (where $\mathsf{DP}_u(\mathcal{R}, \mu) = \varnothing$ whenever $\mathsf{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \varnothing$). Then $\mathsf{DP}'(\mathcal{R}, \mu) = \mathsf{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \mathsf{DP}_u(\mathcal{R}, \mu)$.

The corresponding definition of *chain* is, essentially, the standard one [10], but μ-rewriting (with $\mathcal{R}$) is used for connecting pairs.

Definition 19 [1, Definition 11]. Let $\mathcal{P}$ and $\mathcal{R}$ be TRSs and let μ be a replacement map. We extend μ to tuple symbols by defining $\mu(f^\sharp) = \mu(f)$ for all $f \in \mathcal{D}$ and $\mu(\mathtt{U}) = \varnothing$. A sequence of pairs $u_1 \to v_1, u_2 \to v_2, \ldots$ from $\mathcal{P}$ is a $(\mathcal{P}, \mathcal{R}, \mu)$-*chain* if there is a substitution σ with $\sigma(v_i) \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$ and $\sigma(v_i)$ is $(\mathcal{R}, \mu)$-terminating for all i.

Using these definitions, a characterization of termination of *CSR* is given.

Theorem 17 [1, Theorem 12]. *A TRS $\mathcal{R}$ is μ-terminating if and only if there is no infinite $(\mathsf{DP}'(\mathcal{R}, \mu), \mathcal{R}, \mu)$-chain.*

On the basis of these definitions and results, Alarcón et al. [1, Section 4] develop a CSDP framework.

15.1. Comparing CSDPs and noncollapsing CSDPs

As discussed in Section 14.1, the idea of providing a definition of CSDPs that does not use collapsing pairs cannot be considered as the main contribution of [1]: in 2006 there was an implementation of CSDPs without collapsing pairs (namely the one which corresponds to Definition 16). Actually, Definition 18 is very close to Definition 15 (i.e., the first correct notion of CSDP developed in 2006) if we write $\mathtt{U}$ instead of $\mathtt{MUSUBTERM}$ in Definition 15. The crucial differences between Definition 15 and Definition 18 are the use of *hiding contexts* (Definition 17) and the use of *hidden terms* (Definition 3). As discussed in [1, Section 3], the notion of hiding context is a refinement of the notion of hidden term described in this paper (and previously approached in [4]), see Section 3.2.

Indeed, the notion of hiding context is the most important contribution of [1] from the theoretical side. The notion of hiding context can be easily integrated in the CSDP framework discussed in this paper. This has been carried out in [27,28,37], where an extension of our CSDP framework was developed to appropriately integrate this notion. Within this new approach, Definition 18 could be incorporated to the CSDP framework by using the following modified version of Theorem 8, which defines the appropriate *CS processor*.

Theorem 18. *Let $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}}$. Let $u \to x \in \mathcal{P}_{\mathcal{X}}$ and*

$$\begin{aligned} P_u = {} & \{u \to U(x)\} \\ & \cup \{U(f(x_1, \ldots, x_k)) \to U(x_i) \mid f \in \mathcal{F}, 1 \leqslant i \leqslant ar(f) \text{ and } f \text{ hides } i\} \\ & \cup \{U(t) \to t^\sharp \mid t \in \mathcal{NHT}_{\mathcal{P}}\} \end{aligned}$$

where U is a fresh symbol. Let $\mathcal{P}' = (\mathcal{G} \cup \{U\}, P')$, where $P' = (P - \{u \to x\}) \cup P_u$, and μ' which extends μ by $\mu'(U) = \varnothing$. Then, the processor Proc_{hCtx} given by

$$\mathsf{Proc}_{hCtx}(\mathcal{P}, \mathcal{R}, \mu) = \{(\mathcal{P}', \mathcal{R}, \mu')\}$$

is sound and complete.

The proof of this result would be analogous to the one for Theorem 8 with the proviso, in our definition of chain of pairs (Definition 5), that the contexts $C_i[\ ]_{p_i}$ which are used for handling collapsing pairs are now *hiding contexts*. In contrast to

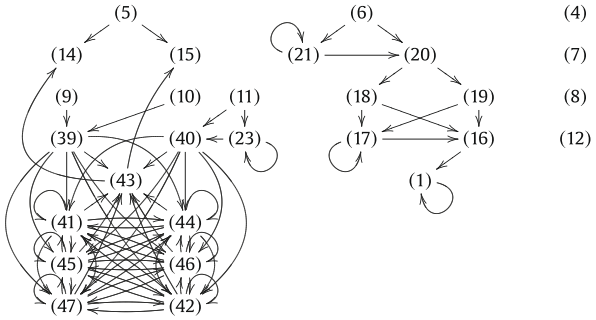


Fig. 7. Context-sensitive dependency graph of Example 1 according to [1].

processor Proc_{eColl} in Theorem 8, Proc_{hCtx} has the advantage of introducing fewer rules due to the use of the notion of *hiding* in Definition 17. Obviously, this could lead to simpler proofs when it is used.

In this paper, we have shown that collapsing pairs are an essential part of the theoretical description of termination of *CSR*. Actually, Definition 18 *explicitly* uses them to introduce the new unhiding pairs. This shows that the most basic notion when *modeling* the termination behavior of *CSR* is that of collapsing pair and that unhiding pairs should be better considered as an ingredient for handling collapsing pairs in proofs of termination (as implemented by processor Proc_{hCtx} above).

15.2. Use of CSDPs and noncollapsing CSDPs

The application of Definition 18 at the very beginning of the termination analysis of CS-TRSs (as done in [1]) often leads to obtaining a more complex dependency graph. For instance, we would *replace* the collapsing CSDPs (25) and (26) by the following ones:

$$\mathtt{TAIL}(\mathtt{cons}(x, xs)) \to \mathtt{U}(xs) \tag{39}$$

$$\mathtt{TAKE}(\mathtt{s}(n), \mathtt{cons}(x, xs)) \to \mathtt{U}(xs) \tag{40}$$

$$\mathtt{U}(\mathtt{incr}(x)) \to \mathtt{U}(x) \tag{41}$$

$$\mathtt{U}(\mathtt{incr}(\mathtt{oddNs})) \to \mathtt{INCR}(\mathtt{oddNs}) \tag{42}$$

$$\mathtt{U}(\mathtt{oddNs}) \to \mathtt{ODDNS} \tag{43}$$

$$\mathtt{U}(\mathtt{rep2}(x)) \to \mathtt{U}(x) \tag{44}$$

$$\mathtt{U}(\mathtt{zip}(x, y)) \to \mathtt{U}(x) \tag{45}$$

$$\mathtt{U}(\mathtt{zip}(x, y)) \to \mathtt{U}(y) \tag{46}$$

$$\mathtt{U}(\mathtt{cons}(x, y)) \to \mathtt{U}(x) \tag{47}$$

to obtain the graph in Fig. 7, which should be compared with the CSDG for the same example in Fig. 5. On the other hand, if $\mathcal{P}$ contains no collapsing pairs (as happens if Definition 18 is used to compute the dependency pairs of a CS-TRS), then Definition 19 is subsumed by our notion of chain of pairs (Definition 5). This means that, after using processor Proc_{eColl} in Theorem 8 to remove collapsing pairs in the component $\mathcal{P}$ of a CS problem $(\mathcal{P}, \mathcal{R}, \mu)$, *we could use all CS processors developed in [1]*, some of which have not been discussed in our paper (for instance, the *instantiation processor* [1, Theorem 24]). Also, the CS processors that are developed here can be used in any implementation following [1].

Remark 16. Note that, although the definition of chain in [1] (see Definition 19) is apparently closer to the standard one [35, Definition 3], this does *not* mean that we can use or easily 'translate' existing DP-processors (see [35]) to be used with *CSR*.

The narrowing processor provides a striking example. Example 29 shows that the application of the narrowing processor to the TRSs $\mathcal{P}$ and $\mathcal{R}$ in the example is *not* correct due to the lack of *strong* μ*-conservativeness* of the μ-narrowed pair in $\mathcal{P}$. Since $\mathcal{P}$ has no collapsing pair, one could think (following a naïve interpretation of [1]) that the narrowing processor of the DP-framework (see [35, Theorem 31]), which does not take into account the replacement restrictions, should work with *CSR* without difficulties, which is not the case.

Table 3
Summary of processors used in MU-TERM-IC.

Processors	Applied in
SCC processor (Section 8)	94/94 problems
Processors based on subterm criteria (Section 11)	56/94 problems
Processors based on reduction pairs (Section 10)	50/94 problems
Basic processors (Section 7)	28/94 problems
Narrowing processor (Section 12)	3/94 problems

Thus, a CSDP framework that is based on Definitions 18 and 19 *does not boil down to the DP-framework*, and a careful consideration of the replacement restrictions is necessary before being able to use any DP-processor with *CSR*.

15.3. Experimental evaluation

We have performed an experimental evaluation of the use of CSDPs vs. the ones in Definition 18 as follows: we prepared two versions of MU-TERM: MU-TERM-LPAR08 and MU-TERM-IC. The tool MU-TERM-LPAR08 first applies Theorem 18 to remove all collapsing pairs (as one would do when working within the approach described in [1]) and then uses the CS processors described in both this paper and in [1] to achieve termination proofs. On the other hand, MU-TERM-IC implements the CSDP framework that we have described here (with the modifications developed in [27,28,37]).

On a collection of 109 examples, both tools succeeded on the very same ones (94 proofs of termination). However, MU-TERM-IC performed globally faster. Furthermore, we *did not need to use* Proc_{hCtx} in the proofs with MU-TERM-IC. This suggests that (in contrast to what we claimed in [1] when the integration of the notion of hiding context into the CSDP framework was pending), collapsing pairs do not represent any drawback for automatically proving termination of *CSR*. Detailed benchmarks are at the following URL: http://zenon.dsic.upv.es/muterm/benchmarks/ic10/muterm-2009/benchmarks.html Table 3 shows the use of the different processors in these benchmarks. The interpretation of the frequency of use for the different processors should take into account the following *strategy* for invoking them in MU-TERM-IC when CS problems are treated: first, we try the basic (infinite and finite) processors. If some of them succeed, we are done; otherwise, we continue as follows:

1. SCC processor.
2. Subterm criterion processors.
3. Reduction pair (RP) processors with polynomial and matrix interpretations over the reals [6,7,49,51].
4. Narrowing processor.

Interestingly, *all* processors are used at least once during the proofs.

16. Conclusions

We have analyzed the structure of infinite context-sensitive rewrite sequences starting from minimal non-μ-terminating terms (Theorem 1). This knowledge is used to provide an appropriate definition of context-sensitive dependency pair (Definition 4), and the related notion of chain (Definition 5). In sharp contrast to the standard dependency pair approach, where all dependency pairs have tuple symbols $f^\sharp$ in both the left- and right-hand sides, we have *collapsing* dependency pairs that have a single *variable* in the right-hand side. These variables reflect the effect of the *migrating* variables on the termination behavior of *CSR*. At the level of *minimal chains*, however, the contrast with the ordinary DP approach is somehow recovered by a nice symmetry arising from the central notion of *hidden term* (Definition 3): a noncollapsing pair $u \to v$ is followed by a pair $u' \to v'$ if $\sigma(v)$ μ-rewrites into $\sigma(u')$ for some substitution σ; a collapsing pair $u \to v$ is followed by a pair $u' \to v'$ if there is a *hidden term* t such that $\sigma(t)^\sharp$ μ-rewrites into $\sigma(u')$ for some substitution σ. We have shown how to use the context-sensitive dependency pairs in proofs of termination of *CSR*. As in Arts and Giesl's approach, the absence of infinite minimal chains of dependency pairs from $\mathsf{DP}(\mathcal{R}, \mu)$ characterizes the μ-termination of $\mathcal{R}$ (Theorems 2 and 3).

We have provided a suitable adaptation of the *dependency pair framework* to *CSR* by defining appropriate notions of *CS problem* (Definition 6) and *CS processor* (Definition 7). We have described a number of sound and (most of them) complete CS processors that can be used in any practical implementation of the CSDP-framework. In particular, we have introduced the notion of (estimated) *context-sensitive (dependency) graph* (Definitions 8 and 10) and the associated CS processor (Theorem 6). We have also described some CS processors for removing or transforming collapsing pairs from CS problems (Theorems 7 and 8). We are also able to use μ-reduction pairs (Definition 11) and argument filterings to ensure the absence of infinite chains of pairs (Theorems 9, 10, and 11). We have adapted Hirokawa and Middeldorp's *subterm criterion* which permits concluding the absence of infinite minimal chains by paying attention only to the pairs in the corresponding CS problem (Theorems 12 and 13). Following this appealing idea, we have also introduced two new processors that work in a similar way but use a very basic kind of ordering instead of the subterm relation (Theorems 14 and 15). Narrowing context-sensitive dependency pairs have also been investigated. It is helpful to simplify or restructure the dependency graph and eventually simplify the proof of termination (Theorem 16).

We have implemented these ideas as part of the termination tool MU-TERM [2,47]. The implementation and practical use of the developed techniques yield a novel and powerful framework that improves the current state-of-the-art of methods for proving termination of *CSR*. Actually, CSDPs were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007 competition of termination tools.

For future work, we plan to extend the basic CSDP-framework described in this paper with further CS processors integrating the *usable rules* for *CSR* [29] and proofs of termination of *innermost CSR* using CSDPs [5].

Acknowledgments

We thank Jürgen Giesl and his group of the RWTH Aachen (especially Fabian Emmes, Carsten Fuhs, Peter Schneider-Kamp, and René Thiemann) for many fruitful discussions about CSDPs. We also thank the anonymous referees for many useful remarks.

Appendix A

A.1. Proofs of Section 3

Lemma 2. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If s is not μ-terminating, then there is a subterm t of s ($s \trianglerighteq t$) such that $t \in \mathcal{T}_{\infty,\mu}$.*

Proof. By structural induction. If s is a constant symbol, it is obvious: take $t = s$. If $s = f(s_1, \ldots, s_k)$, then we proceed by contradiction. If there is no subterm t of s such that $t \in \mathcal{T}_{\infty,\mu}$, then $s \notin \mathcal{T}_{\infty,\mu}$. Since s is not μ-terminating, there is a strict subterm t of s ($s \triangleright t$) that is not μ-terminating. By the Induction Hypothesis, there is $t' \in \mathcal{T}_{\infty,\mu}$ such that $t \trianglerighteq t'$. Then, we have $s \triangleright t'$, thus leading to a contradiction. □

Lemma 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If s is not μ-terminating, then there is a μ-replacing subterm t of s such that $t \in \mathcal{M}_{\infty,\mu}$.*

Proof. By structural induction. If s is a constant symbol, it is obvious: take $t = s$. If $s = f(s_1, \ldots, s_k)$, then we proceed by contradiction. If there is no μ-replacing subterm t of s such that $t \in \mathcal{M}_{\infty,\mu}$, then $s \notin \mathcal{M}_{\infty,\mu}$, i.e., there is a strict μ-replacing subterm t of s which is not μ-terminating. We have $t = s|_p$ for some $p \in \mathcal{P}os^{\mu}(s) - \{\Lambda\}$. By the Induction Hypothesis, t contains a μ-replacing subterm t' which belongs to $\mathcal{M}_{\infty,\mu}$, i.e., $t' = t|_q$ for some $q \in \mathcal{P}os^{\mu}(t)$. By Proposition 1, $p \cdot q \in \mathcal{P}os^{\mu}(s)$. Thus, t' is a μ-replacing subterm of s that belongs to $\mathcal{M}_{\infty,\mu}$, thus leading to a contradiction. □

Lemma 4. *Let $\mathcal{R}$ be a TRS, $\mu \in M_{\mathcal{R}}$, and $t \in \mathcal{M}_{\infty,\mu}$. If $t \overset{>\Lambda}{\hookrightarrow}^* u$ and u is non-μ-terminating, then $u \in \mathcal{M}_{\infty,\mu}$.*

Proof. All μ-rewritings below of t the root are issued on μ-replacing and μ-terminating terms that remain μ-terminating by Lemma 1. Then, all strict μ-replacing subterms of u (which are the ones that can be originated or transformed by μ-rewritings from t to u) are μ-terminating. Since u is non-μ-terminating, $u \in \mathcal{M}_{\infty,\mu}$. □

16.1. Proofs of Section 3.2

Lemma 5. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution. If there is a rule $l \to r \in R$ such that $\sigma(l) \not\trianglerighteq t$ and $\sigma(r) \rhd_{\not\mu} t$, then there is no $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$. Furthermore, there is a term $t' \in \mathcal{HT}$ such that $r \rhd_{\not\mu} t'$ and $\sigma(t') = t$.*

Proof. By contradiction. If there is $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$, then since variables in l are always below some function symbol we have $\sigma(l) \triangleright t$, leading to a contradiction.

Since there is no $x \in \mathcal{V}ar(r)$ such that $\sigma(x) \trianglerighteq t$ but we have that $\sigma(r) \rhd_{\not\mu} t$, then there is a nonvariable and non-μ-replacing position $p \in \mathcal{P}os_{\mathcal{F}}(r) - \mathcal{P}os^{\mu}(r)$ of r, such that $\sigma(r|_p) = t$. Then, we let $t' = r|_p$. Note that $t' \in \mathcal{HT}$. □

Lemma 6. *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let A be a μ-rewrite sequence $t_1 \hookrightarrow t_2 \hookrightarrow \cdots \hookrightarrow t_n$ with $t_i \in \mathcal{M}_{\infty,\mu}$ for all i, $1 \leqslant i \leqslant n$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_1 \not\trianglerighteq t$ and $t_n \rhd_{\not\mu} t$, then $t = \sigma(s)$ for some $s \in \mathcal{DHT}$ and substitution σ.*

Proof. By induction on n:

1. If $n = 1$, then it is vacuously true because $t_1 \not\trianglerighteq t$ and $t_1 \rhd_{\not\mu} t$ do not simultaneously hold.

2. If $n > 1$, then we assume that $t_1 \not\rhd t$ and $t_n \rhd_\mu t$. We consider two cases:
   (a) If $t_{n-1} \rhd_\mu t$, then by the induction hypothesis the conclusion follows.
   (b) If $t_{n-1} \rhd_\mu t$ does not hold, then, since assuming $t_{n-1} \rhd_\mu t$ leads to a contradiction (because $t_{n-1} \in \mathcal{M}_{\infty,\mu}$ in the hypothesis implies that $t \notin \mathcal{M}_{\infty,\mu}$), we have that $t_{n-1} \not\rhd t$. Let $l \to r \in R$ be such that $t_{n-1} = C[\sigma(l)]$ and $t_n = C[\sigma(r)]$ for some context $C[\,]$ and substitution σ. Then, in particular, $\sigma(l) \not\rhd t$ and, since $t_n \rhd_\mu t$ there must be $\sigma(r) \rhd_\mu t$. Thus, by Lemma 5 we conclude that $t = \sigma(s)$ for some $s \in \mathcal{HT}$ and substitution σ. Since $t \in \mathcal{M}_{\infty,\mu}$, it follows that $root(t) = root(s) \in \mathcal{D}$. Thus, $s \in \mathcal{DHT}$. □

Proposition 4. *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. Consider a finite or infinite sequence of the form $t_1 \overset{\Lambda}{\hookrightarrow} s_1 \trianglerighteq_\mu t'_2 \overset{>\Lambda}{\hookrightarrow}{}^* t_2 \overset{\Lambda}{\hookrightarrow} s_2 \trianglerighteq_\mu t'_3 \overset{>\Lambda}{\hookrightarrow}{}^* t_3 \cdots$ with $t_j, t'_j \in \mathcal{M}_{\infty,\mu}$ for all $j \geq 1$. If there is a term $t \in \mathcal{M}_{\infty,\mu}$ such that $t_i \rhd_\mu t$ for some $i \geq 1$, then $t_1 \rhd_\mu t$ or $t = \sigma(s)$ for some $s \in \mathcal{DHT}$ and substitution σ.*

Proof. By induction on i:

1. If $i = 1$, it is trivial.
2. If $i > 1$ and $t_i \rhd_\mu t$, then we consider two cases:
   (a) If $t_{i-1} \rhd_\mu t$, then by the induction hypothesis, the conclusion follows.
   (b) If $t_{i-1} \rhd_\mu t$ does not hold, then let $l \to r \in R$ and σ be such that $t_{i-1} = \sigma(l)$ and $s_{i-1} = \sigma(r) \trianglerighteq_\mu t'_i$. Since $t_{i-1} \rhd_\mu t$ leads to a contradiction (because $t_{i-1} \in \mathcal{M}_{\infty,\mu}$ implies that $t \notin \mathcal{M}_{\infty,\mu}$), we have that $t_{i-1} \not\rhd t$. We consider two cases:
      (A) If $t'_i \rhd t$, then, since $t'_i, t \in \mathcal{M}_{\infty,\mu}$, the case $t'_i \rhd_\mu t$ is excluded and the only possibility is that $t'_i \rhd_\mu t$. Then, since $\sigma(l) = t_{i-1} \not\rhd t$ and $\sigma(r) \trianglerighteq_\mu t'_i \rhd_\mu t$, i.e., $\sigma(r) \rhd_\mu t$, by Lemma 5 we conclude that $t = \sigma(s)$ for some $s \in \mathcal{HT}$ and substitution σ. Since $t \in \mathcal{M}_{\infty,\mu}$, it follows that $root(t) = root(s) \in \mathcal{D}$. Thus, $s \in \mathcal{DHT}$.
      (B) If $t'_i \not\rhd t$, then, by applying Lemma 4 and Lemma 6 to the μ-rewrite sequence $t'_i \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathcal{R},\mu} t_i$, the conclusion follows. □

References

[1] B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, R. Thiemann, Improving context-sensitive dependency pairs, in: I. Cervesato, H. Veith, A. Voronkov (Eds.), Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08, LNAI, vol. 5330, Springer, Berlin, 2008, pp. 636–651.
[2] B. Alarcón, R. Gutiérrez, J. Iborra, S. Lucas, Proving termination of context-sensitive rewriting with MU-TERM, Electronic Notes in Theoretical Computer Science 188 (2007) 105–115.
[3] B. Alarcón, R. Gutiérrez, S. Lucas, Context-sensitive dependency pairs, in: S. Arun-Kumar, N. Garg (Eds.), Proceedings of the XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06, LNCS, vol. 4337, Springer, Berlin, 2006, pp. 297–308.
[4] B. Alarcón, R. Gutiérrez, S. Lucas, Improving the context-sensitive dependency graph, Electronic Notes in Theoretical Computer Science 188 (2007) 91–103.
[5] B. Alarcón, S. Lucas, Termination of innermost context-sensitive rewriting using dependency pairs, in: B. Konev, F. Wolter (Eds.), Proceedings of the Sixth International Symposium on Frontiers of Combining Systems, FroCoS'07, LNAI, vol. 4720, Springer, Berlin, 2007, pp. 73–87.
[6] B. Alarcón, S. Lucas, R. Navarro-Marset, Proving termination with matrix interpretations over the reals, in: A. Geser, J. Waldmann (Eds.), Proceedings of the 10th International Workshop on Termination, WST'09, June 2009, pp. 12–15.
[7] B. Alarcón, S. Lucas, R. Navarro-Marset, Using matrix interpretations over the reals in proofs of termination, in: F. Lucio, G. Moreno, R. Peña (Eds.), Proceedings of the Ninth Spanish Conference on Programming and Computer Languages, PROLE'09, Universidad del País Vasco, 2009, pp. 255–264.
[8] M. Alpuente, S. Escobar, B. Gramlich, S. Lucas, On-demand strategy annotations revisited: an improved on-demand evaluation strategy, Theoretical Computer Science 411 (2) (2010) 504–541.
[9] T. Arts, Automatically proving termination and innermost normalisation of term rewriting systems, Ph.D. Thesis, Universiteit Utrecht, 1997.
[10] T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theoretical Computer Science 236 (1–2) (2000) 133–178.
[11] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, Cambridge, 1998.
[12] C. Borralleras, S. Lucas, A. Rubio, Recursive path orderings can be context-sensitive, in: A. Voronkov (Ed.), Proceedings of the XVIII Conference on Automated Deduction, CADE'02, LNAI, vol. 2392, Springer, Berlin, 2002, pp. 314–331.
[13] R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theoretical Computer Science 351 (1) (2006) 386–414.
[14] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, C. Talcott. All about Maude – a high-performance logical framework, in: Lecture Notes in Computer Science, vol. 4350, Springer, Berlin, 2007.
[15] N. Dershowitz, Termination by abstraction, in: B. Demoen, V. Lifschitz (Eds.), Proceedings of the 20th International Conference on Logic Programming, ICLP'04, LNCS, vol. 3132, Springer, Berlin, 2004, pp. 1–18.
[16] F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving termination of membership equational programs, Proceedings of the 2004 ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation, PEPM'04, ACM Press, New York, 2004, pp. 147–158.
[17] F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving operational termination of membership equational programs, Higher-Order and Symbolic Computation 21 (1–2) (2008) 59–88.
[18] J. Endrullis, Jambox: Automated termination proofs for string and term rewriting. Available from: <http://joerg.endrullis.de/jambox.html>.
[19] J. Endrullis, D. Hendriks, From outermost to context-sensitive rewriting, in: R. Treinen (Ed.), Proceedings of the 20th International Conference on Rewriting Techniques and Applications, RTA'09, LNCS, vol. 5595, Springer, Berlin, 2009, pp. 305–319.
[20] J. Endrullis, J. Waldmann, H. Zantema, Matrix interpretations for proving termination of term rewriting, Journal of Automated Reasoning 40 (2–3) (2008) 195–220.
[21] M.-L. Fernández, Relaxing monotonicity for innermost termination, Information Processing Letters 93 (2005) 117–123.
[22] M.C.F. Ferreira, A.L. Ribeiro, Context-sensitive AC-rewriting, in: P. Narendran, M. Rusinowitch (Eds.), Proceedings of the 10th International Conference on Rewriting Techniques and Applications, RTA'99, LNCS, vol. 1631, Springer, Berlin, 1999, pp. 286–300.

[23] K. Futatsugi, J. Goguen, J.-P. Jouannaud, J. Meseguer, Principles of OBJ2, in: Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85, ACM Press, New York, 1985, pp. 52–66.
[24] K. Futatsugi, A. Nakagawa. an overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specification over networks, in: Proceedings of the First International Conference on Formal Engineering Methods, 1997.
[25] J. Giesl, T. Arts, E. Ohlebusch, Modular termination proofs for rewriting using dependency pairs, Journal of Symbolic Computation 34 (1) (2002) 21–58.
[26] B. Gramlich, S. Lucas, Simple termination of context-sensitive rewriting, in: B. Fischer, E. Visser (Eds.), Proceedings of the III ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02, ACM Press, New York, 2002, pp. 29–41.
[27] R. Gutiérrez, S. Lucas, Mechanizing proofs of termination in the context-sensitive dependency pairs framework, in: F. Lucio, G. Moreno, R. Peña (Eds.), Proceedings of the Ninth Spanish Conference on Programming and Computer Languages, PROLE'09, Universidad del País Vasco, 2009, pp. 265–274.
[28] R. Gutiérrez, S. Lucas, Mechanizing proofs of termination with context-sensitive dependency pairs, in: A. Geser, J. Waldmann (Eds.), Proceedings of the 10th International Workshop on Termination, WST'09, HTWK Leipzig, 2009, pp. 43–46.
[29] R. Gutiérrez, S. Lucas, X. Urbain, Usable rules for context-sensitive rewrite systems, in: A. Voronkov (Ed.), Proceedings of the 19th International Conference on Rewriting Techniques and Applications, RTA'08, LNCS, vol. 5117, Springer, Berlin, 2008, pp. 126–141.
[30] J. Giesl, A. Middeldorp, Transforming context-sensitive rewrite systems, in: P. Narendran, M. Rusinowitch (Eds.), Proceedings of the X International Conference on Rewriting Techniques and Applications, RTA'99, LNCS, vol. 1631, Springer, Berlin, 1999, pp. 271–285.
[31] J. Giesl, A. Middeldorp, Transformation techniques for context-sensitive rewrite systems, Journal of Functional Programming 14 (4) (2004) 379–427.
[32] J. Giesl, P. Schneider-Kamp, R. Thiemann, AProVE 1.2: automatic termination proofs in the dependency pair framework, in: U. Furbach, N. Shankar (Eds.), Proceedings of the Third International Joint Conference on Automated Reasoning, IJCAR'06, LNAI, vol. 4130, Springer, Berlin, 2006, pp. 281–286. Available from: <http://aprove.informatik.rwth-aachen.de/>.
[33] J. Giesl, R. Thiemann, P. Schneider-Kamp, The dependency pair framework: combining techniques for automated termination proofs, in: F. Baader, A. Voronkov (Eds.), Proceedings of the XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04, LNAI, vol. 3452, Springer, Berlin, 2004, pp. 301–331.
[34] J. Giesl, R. Thiemann, P. Schneider-Kamp, Proving and disproving termination of higher-order functions, in: B. Gramlich (Ed.), Proceedings of the Fifth International Workshop on Frontiers of Combining Systems, FroCoS'05, LNAI, vol. 3717, Springer, Berlin, 2005, pp. 216–231.
[35] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Mechanizing and improving dependency pairs, Journal of Automatic Reasoning 37 (3) (2006) 155–203.
[36] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, J.-P. Jouannaud, Introducing OBJ, in: J. Goguen, G. Malcolm (Eds.), Software Engineering with OBJ: Algebraic Specification in Action, Kluwer, Dordrecht, 2000, pp. 1–2.
[37] R. Gutiérrez, Context-sensitive dependency pairs framework, Master's thesis, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, València, Spain, December 2008.
[38] N. Hirokawa, A. Middeldorp, Dependency pairs revisited, in: V. van Oostrom (Ed.), Proceedings of the XV International Conference on Rewriting Techniques and Applications, RTA'04, LNCS, vol. 3091, Springer, Berlin, 2004, pp. 249–268.
[39] N. Hirokawa, A. Middeldorp, Automating the dependency pair method, Information and Computation 199 (2005) 172–199.
[40] N. Hirokawa, A. Middeldorp, Tyrolean termination tool: techniques and features, Information and Computation 205 (2007) 474–511.
[41] P. Hudak, S.J. Peyton-Jones, P. Wadler, Report on the functional programming language Haskell: a non-strict, purely functional language, Sigplan Notices 27 (5) (1992) 1–164.
[42] K. Kusakari, M. Nakamura, Y. Toyama, Argument filtering transformation, in: G. Nadathur (Ed.), Proceedings of the International Conference on Principles and Practice of Declarative Programming, PPDP'99, LNCS, vol. 1702, 1999, pp. 47–61.
[43] S. Lucas. Termination of context-sensitive rewriting by rewriting, in: F. Meyer auf der Heide, B. Monien (Eds.), Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, ICALP'96, LNCS, vol. 1099, Springer, Berlin, 1996, pp. 122–133.
[44] S. Lucas, Context-sensitive computations in functional and functional logic programs, Journal of Functional and Logic Programming 1998 (1) (1998) 1–61.
[45] S. Lucas, Termination of on-demand rewriting and termination of OBJ programs, in: Proceedings of the Third International Conference on Principles and Practice of Declarative Programming, PPDP'01, ACM Press, New York, 2001, pp. 82–93.
[46] S. Lucas, Context-sensitive rewriting strategies, Information and Computation 178 (1) (2002) 293–343.
[47] S. Lucas, MU-TERM: a tool for proving termination of context-sensitive rewriting. in: V. van Oostrom (Ed.), Proceedings of the XV International Conference on Rewriting Techniques and Applications, RTA'04, LNCS, vol. 3091, Springer, Berlin, 2004, pp. 200–209. Available from: <http://zenon.dsic.upv.es/muterm/>.
[48] S. Lucas, Polynomials for proving termination of context-sensitive rewriting, in: I. Walukiewicz (Ed.), Proceedings of the VII International Conference on Foundations of Software Science and Computation Structures, FOSSACS'04, LNCS, vol. 2987, Springer, Berlin, 2004, pp. 318–332.
[49] S. Lucas, Polynomials over the reals in proofs of termination: from theory to practice, RAIRO Theoretical Informatics and Applications 39 (3) (2005) 547–586.
[50] S. Lucas, Proving Termination of Context-Sensitive Rewriting by Transformation, Information and Computation 204 (12) (2006) 1782–1846.
[51] S. Lucas, Practical use of polynomials over the reals in proofs of termination, in: Proceedings of the Ninth International Symposium on Principles and Practice of Declarative Programming, PPDP'07, ACM Press, New York, 2007, pp. 39–50.
[52] S. Lucas, J. Meseguer, Operational termination of membership equational programs: the order-sorted way, Electronic Notes in Theoretical Computer Science 238 (3) (2009) 207–225.
[53] S. Lucas, J. Meseguer, Order-sorted dependency pairs, in: Proceedings of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08, ACM Press, New York, 2008, pp. 108–119.
[54] J. McCarthy, Recursive functions of symbolic expressions and their computations by machine. Part I, Communications of the ACM 3 (4) (1960) 184–195.
[55] A. Middeldorp, Approximating dependency graphs using tree automata techniques. in: R. Goré, A. Leitsch, T. Nipkow (Eds.), Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'01, LNAI, vol. 2083, 2001, pp. 593–610.
[56] A. Middeldorp, Approximations for strategies and termination, Electronic Notes in Computer Science 70 (6) (2002).
[57] E.G.J.M.H. Nöcker, J.E.W. Smetsers, M.C.J.D. van Eekelen, M.J. Plasmeijer, Concurrent clean, in: E.H.L. Aarts, J. Leeuwen, M. Rem (Eds.), Proceedings of the Parallel Architectures and Languages Europe, PARLE'91, LNCS, vol. 506, Springer, Berlin, 1992, pp. 202–219.
[58] E. Ohlebusch, Advanced Topics in Term Rewriting, Springer, Berlin, 2002.
[59] F. Schernhammer, B. Gramlich, Termination of lazy rewriting revisited, Electronic Notes in Theoretical Computer Science 204 (2008) 35–51.
[60] F. Schernhammer, B. Gramlich, VMTL-A modular termination laboratory, in: R. Treinen (Ed.), Proceedings of the 20th International Conference on Rewriting Techniques and Applications, RTA'09, LNCS, vol. 5595, Springer, Berlin, 2009, pp. 285–294.
[61] TeReSe, Term Rewriting Systems, Cambridge University Press, Cambridge, 2003.
[62] R. Thiemann, The DP framework for proving termination of term rewriting, Ph.D. Thesis. Available as Technical Report AIB-2007-17, RWTH Aachen, Germany, 2007.
[63] H. Zantema, Termination of context-sensitive rewriting, in: H. Comon (Ed.), Proceedings of the VII International Conference on Rewriting Techniques and Applications, RTA'97, LNCS, vol. 1232, Springer, Berlin, 1997, pp. 172–186.

18.8 A Dependency Pair Framework for $A \vee C$-Termination

8. B. Alarcón, S. Lucas, and J. Meseguer. **A Dependency Pair Framework for $A \vee C$-Termination**. In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 35–51. Springer-Verlag, 2010.

A Dependency Pair Framework for $A \vee C$-Termination*

Beatriz Alarcón[1], Salvador Lucas[1], and José Meseguer[2]

[1] ELP group, DSIC, Universidad Politécnica de Valencia,
Camino de Vera s/n, 46022 Valencia, Spain
{balarcon,slucas}@dsic.upv.es
[2] CS Dept. University of Illinois at Urbana-Champaign, IL, USA

Abstract. The development of powerful techniques for proving termination of rewriting modulo a set of equations is essential when dealing with rewriting logic-based programming languages like CafeOBJ, Maude, OBJ, etc. One of the most important techniques for proving termination over a wide range of variants of rewriting (strategies) is the *dependency pair approach*. Several works have tried to adapt it to rewriting modulo *associative and commutative* (AC) equational theories, and even to more general theories. However, as we discuss in this paper, no appropriate notion of minimality (and minimal chain of dependency pairs) which is well-suited to develop a *dependency pair framework* has been proposed to date. In this paper we carefully analyze the structure of infinite rewrite sequences for rewrite theories whose equational part is a (free) *combination* of associative and commutative axioms which we call $A \vee C$*-rewrite theories*. Our analysis leads to a more accurate and optimized notion of dependency pairs through the new notion of *stably minimal term*. Then, we have developed a suitable dependency pair framework for proving termination of $A \vee C$-rewrite theories.

Keywords: equational rewriting, termination, dependency pairs.

1 Introduction

Rewriting with rules R modulo axioms E is a widely used technique in both rule-based programming languages and in automated deduction. Consequently, termination of rewriting modulo specific axioms E (e.g., associativity-commutativity, AC) has been studied. Methods for proving termination of rewriting systems modulo AC-axioms are known and even implemented. Several works have tried to adapt the *dependency pair approach* (DP-approach [1]) to rewriting modulo *associative and commutative* (AC) theories [13,9,10,11,14]. The corresponding proof methods, though, cannot be applied to commonly occurring combinations

* Partially supported by EU (FEDER) and MICINN grant TIN 2007-68093-C02-02. José Meseguer has been partially supported by NSF Grants CCF-0905584, CNS-07-16038, and CNS-08-34709. Beatriz Alarcón was partially supported by the Spanish MEC/MICINN under FPU grant AP2005-3399.

P.C. Ölveczky (Ed.): WRLA 2010, LNCS 6381, pp. 35–51, 2010.
© Springer-Verlag Berlin Heidelberg 2010

```
fmod LIST&SET is
  sorts Bool Nat List Set .
  subsorts Nat < List Set .
  ops true false : -> Bool .
  ops _and_ _or_ : Bool Bool -> Bool [assoc comm] .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
  op _;_ : List List -> List [assoc] .
  op null : -> Set .
  op __ : Set Set -> Set [assoc comm] .
  op _in_ : Nat Set -> Bool .
  op _==_ : List List -> Bool [comm] .
  op list2set : List -> Set .
  var  B : Bool .              vars N M : Nat .
  vars L L' : List .           var  S : Set .
  eq N N = N .
  eq true and B = B .          eq false and B = false .
  eq true or B = true .        eq false or B = B .
  eq 0 == s N = false .        eq s N == s M = N == M .
  eq N ; L == M = false .      eq N ; L == M ; L' = (N == M) and L == L' .
  eq L == L = true .
  eq list2set(N) = N .         eq list2set(N ; L) = N list2set(L) .
  eq N in null = false .       eq N in M S = (N == M) or N in S .
endfm
```

Fig. 1. Example in Maude syntax [3]

of axioms that fall outside their scope. For instance, they could not be applied to prove termination of the TRS in Figure 1, (specified in Maude with self-explanatory syntax; we would not care about sort information here) where we have a (free) *combination* of associative and commutative axioms which we call an *$A \vee C$-rewrite theory* in this paper. Furthermore, the *Dependency Pair Framework* (DP-framework [6]), which is the basis of state-of-the-art tools for proving termination of (different variants of) term rewriting has not yet been adapted to the AC case.

In this paper, we address these two problems. Giesl and Kapur generalized the previous works on AC-termination with dependency pairs to deal with more general kinds of equational theories E satisfying some restrictions [5]. In principle, the $A \vee C$-theories that we are going to investigate here fit Giesl and Kapur's approach. However, as we discuss below, they did not provide any definition of *minimal chain* needed for further developments in the DP-framework. In the DP-framework, the central notion regarding termination proofs is that of *DP problem*: the goal is checking the absence (or presence) of the so-called *infinite minimal chains*, where the notion of minimal chain can be thought as an abstraction of the infinite rewrite sequences starting from *minimal non-terminating terms*. The most important notion regarding mechanization of the proofs is that

of *processor*. A (correct) processor basically transforms DP problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. Here 'simpler' usually means that fewer pairs are involved. Processors are used in a pipe (more precissely, a *tree*) to incrementally simplify the original DP problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. This is the crucial new feature of the DP-framework w.r.t. the DP-approach of [1]. This makes it so powerful as a basis for implementing termination provers.

Before being able to adapt the DP-framework to deal with $A \vee C$-theories, we start by giving a more refined notion of minimality. In fact, the notion of minimality which is used in [5] is the straightforward extension of the one which is used to prove termination of standard rewriting but without dealing with *equivalence preservation* which, as we show below, is essential to provide an appropriate notion of minimal non-E-terminating term for $A\!\vee\!C$-theories E which can be used to define a suitable $A \vee C$-DP-framework. We carefully analyze the structure of infinite rewrite sequences for $A \vee C$-rewrite theories. This leads to appropriate definitions of $A \vee C$-dependency pair and minimal chain.

After some technical preliminaries, in Section 3 we investigate the drawbacks of previous notions of minimal term when modeling infinite $A \vee C$-rewrite sequences. Then, we introduce the notion of *stably minimal* non-E-terminating term which is the basis of our development. Section 4 investigates the structure of infinite sequences starting from such stably minimal terms. Section 5 uses these results to formalize our notion of $A\!\vee\!C$-dependency pairs and minimal chains. Section 6 introduces an $A\!\vee\!C$-DP-framework for proving $A\!\vee\!C$-termination using $A\!\vee\!C$-DPs. in particular, we introduce the notion of $A\!\vee\!C$-dependency graph and a first processor for proving termination in the $A\!\vee\!C$-DP-framework. We also show how to use orderings for defining a second processor. Section 7 compares our approach with the related work and concludes.

2 Rewriting Modulo Equational Theories

Given a rewrite theory $\mathcal{R} = (\Sigma, E, R)$, we write $s \to_{R/E} t$ if there exist u, v such that $s \sim_E u$, $u \to_R v$, and $v \sim_E t$. We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is E-*terminating*, iff $\to_{R/E}$ is terminating. In general, given terms s and t, the problem of whether $s \to_{R/E} t$ holds is undecidable: in order to check whether $s \to_{R/E} t$ we have to search through the possibly infinite equivalence classes $[s]_E$ and $[t]_E$ to see whether a matching is found for a subterm of some $u \in [s]_E$ and the result of rewriting u belongs to the equivalence class $[t]_E$. For this reason, a much simpler relation $\to_{R,E}$ is defined, which becomes decidable if an E-matching algorithm exists. For any terms s, t, $s \to_{R,E} t$ holds iff there is a position p in s, a rule $l \to r$ in R, and a substitution σ such that $s|_p \sim_E \sigma(l)$ and $t = s[\sigma(r)]_p$ (see [15]). We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is (R, E)-*terminating*, if $\to_{R,E}$ is terminating.

Regarding E-termination analysis using *dependency pairs* (DPs), Kusakari and Toyama observed that there is no simple extension of DPs to directly deal

with $\rightarrow_{R/E}$-computations [11,9]. In contrast, several approaches have been developed for $\rightarrow_{R,E}$-computations [5,11,13]. Since $\rightarrow_{R,E} \subseteq \rightarrow_{R/E}$ (but the opposite inclusion does not hold, in general), E-termination cannot be concluded from (R,E)-termination. Actually, Marché and Urbain showed that there are (R,E)-terminating rewrite theories $\mathcal{R}$ which are *not* E-terminating.

Example 1. Consider the following rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where '+' is an AC symbol [13]: $a + b \rightarrow a + (b + c)$. Note that $t = a + (b + c)$ is an $\rightarrow_{R,E}$-normal form (hence (R,E)-terminating). However, $t =_{AC} (a + b) + c$ which is non-E-terminating.

Giesl and Kapur [5] proved the equivalence of both notions of termination with respect to a notion of *extension completion* $\mathcal{E}xt_E(R)$ of a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ which, for E being a set containing associative or commutative axioms, goes back to Peterson and Stickel [15].

Theorem 1. [5, Theorem 11] *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory with E a regular and linear equational theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, t starts an infinite $\rightarrow_{R/E}$-reduction if and only if t starts an infinite $\rightarrow_{\mathcal{E}xt_E(R),E}$-reduction. Therefore, $\mathcal{R}$ is E-terminating if and only if $\rightarrow_{\mathcal{E}xt_E(R),E}$ is terminating.*

2.1 Combination of Associative and Commutative Theories

Let E be a set of equations that has the modular decomposition $E = \bigcup_{f \in \Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$, where:

- A_f is the associativity axiom $f(f(x,y),z) = f(x,f(y,z))$,
- C_f is the commutativity axiom $f(x,y) = f(y,x)$.

We also define $\Sigma = \Sigma_A \uplus \Sigma_C \uplus \Sigma_{AC} \uplus \Sigma_\varnothing$ where $f \in \Sigma_A \Leftrightarrow E_f = \{A_f\}$, $f \in \Sigma_C \Leftrightarrow E_f = \{C_f\}$, $f \in \Sigma_{AC} \Leftrightarrow E_f = \{A_f, C_f\}$, $f \in \Sigma_\varnothing \Leftrightarrow E_f = \varnothing$. In the following, we often say that a symbol $f \in \Sigma$ *is associative* if $f \in \Sigma_A \cup \Sigma_{AC}$.

Definition 1 ($A \vee C$-rewrite theory). *An equational theory $E = \bigcup_{f \in \Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$ is called an $A \vee C$-*theory*. A rewrite theory $\mathcal{R} = (\Sigma, E, R)$ such that E is an $A \vee C$-theory, is called an $A \vee C$-*rewrite theory.*

To deal with rewriting modulo $A \vee C$-theories by using (R,E)-rewriting we have to extend R by following [15, Definition 10.4]:

$$\begin{aligned}
\mathcal{E}xt_{AC}(R) &= R \cup \{f(l,w) \rightarrow f(r,w) \mid l \rightarrow r \in R, f = root(l) \in \Sigma_{AC}\} \\
\mathcal{E}xt_{A}(R) &= R \cup \{f(l,w) \rightarrow f(r,w), f(w,l) \rightarrow f(w,r), f(z,f(l,w)) \rightarrow f(z,f(r,w)) \\
&\qquad \mid l \rightarrow r \in R, f = root(l) \in \Sigma_A\} \\
\mathcal{E}xt_{C}(R) &= R
\end{aligned}$$

where w and z are fresh variables which do not occur in the original rule of R. Therefore, given an $A \vee C$ theory E, we let: $\mathcal{E}xt_E(R) = \mathcal{E}xt_{AC}(R) \cup \mathcal{E}xt_A(R) \cup \mathcal{E}xt_C(R)$. Note that $R \subseteq \mathcal{E}xt_E(R)$.

2.2 Minimal Terms and Infinite Rewrite Sequences

Given a TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$, with $\mathcal{C}$ a subsignature of constructors and $\mathcal{D}$ a subsignature of defined symbols, the *minimal* nonterminating terms associated to $\mathcal{R}$ are nonterminating terms t whose proper subterms u (i.e., $t \rhd u$) are terminating; $\mathcal{T}_\infty$ is the set of minimal nonterminating terms associated to $\mathcal{R}$ [7]. Minimal nonterminating terms have two important properties:

1. Every nonterminating term s contains a minimal nonterminating term $t \in \mathcal{T}_\infty$ (i.e., $s \trianglerighteq t$), and
2. minimal nonterminating terms t are always rooted by a *defined* symbol $f \in \mathcal{D}$: $\forall t \in \mathcal{T}_\infty, root(t) \in \mathcal{D}$.

Considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t = f(t_1, \ldots, t_k) \in \mathcal{T}_\infty$ is helpful to arrive at the notion of dependency pair. Such sequences proceed as follows (see, e.g., [7]):

1. a finite number of reductions can be performed *below* the root of t, thus rewriting t into t'; then
2. a rule $f(l_1, \ldots, l_k) \to r$ applies *at the root* of t' (i.e., $t' = \sigma(f(l_1, \ldots, l_k))$ for some substitution σ); and
3. there is a minimal nonterminating term $u \in \mathcal{T}_\infty$ (hence $root(u) \in \mathcal{D}$) at some position p of $\sigma(r)$ which is a *nonvariable position of* r which 'continues' the infinite sequence initiated by t in a similar way.

This means that considering the occurrences of defined symbols in the right-hand sides of the rewrite rules suffices to 'catch' *every possible infinite rewrite sequence starting from* $\sigma(r)$. In particular, no infinite sequence can be issued from t' *below* the variables of r (more precisely: all bindings $\sigma(x)$ are *terminating* terms). The standard definition of dependency pair [1] and (minimal) chain of dependency pairs [6] relies on (1)–(3) above [7]. These facts are formalized as follows:

Proposition 1. [7, Lemma 1] *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS. For all $t \in \mathcal{T}_\infty$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{T}_\infty$ such that $root(u) \in \mathcal{D}$, $t \xrightarrow{>\Lambda}{}^* \sigma(l) \xrightarrow{\Lambda} \sigma(r) \trianglerighteq u$ and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.*

In the following section we begin the analysis of infinite E-rewrite sequences according to this schema. We aim at providing an appropriate notion of minimal non-E-terminating term (for $A \vee C$-theories E) which allows us to reach a result similar to Proposition 1.

3 Stably Minimal Non-E-Terminating Terms

In the dependency pair approach [1,7,6], the analysis of infinite rewrite sequences is restricted to those starting from *minimal nonterminating terms* $t \in \mathcal{T}_\infty$. The following notion of minimal non-E-terminating term is implicit in [5, proof of Theorem 16]. Similar definitions can be found in [10,11,9,14].

Definition 2 (Minimal non-E-terminating term [5]**).** *Let* $\mathcal{R} = (\Sigma, E, R)$ *be a rewrite theory. A non-E-terminating term* $t \in \mathcal{T}(\Sigma, \mathcal{X})$ *is said to be* minimal *(written* $t \in \mathcal{T}_{\infty,R,E}$*) if every strict subterm* s *of* t *(i.e.,* $t \rhd s$*) is* $(\mathcal{E}xt_E(R), E)$*-terminating.*

Remark 1. In Definition 2, if we assume that E is linear and regular (like $A \vee C$-theories), then, by Theorem 1, we could equivalently start by saying that t is non-$(\mathcal{E}xt_E(R), E)$-terminating. This leads to a more symmetric definition which we often use in the following without further comment.

Every non-E-terminating term s contains a minimal non-E-terminating term $t \in \mathcal{T}_{\infty,R,E}$ (this is stated without proof in [5, proof of Theorem 16]).

Remark 2 (Root symbols of minimal terms). Note that, if E is an $A\vee C$-equational theory, then $root(t) \in \mathcal{D}$ whenever $t \in \mathcal{T}_{\infty,R,E}$. As remarked by Giesl and Kapur (see also Example 5 below) this is not true for arbitrary equational theories.

The problem with Giesl and Kapur's Definition 2 is that minimality is *not* preserved under E-equivalence.

Example 2. Consider the following TRS $\mathcal{R}$:

$$f(x,x) \to f(0, f(1,2)) \tag{1}$$

where $f \in \Sigma_{AC}$. Hence, $\mathcal{E}xt_{AC}(R)$ only adds the following rule to $\mathcal{R}$:

$$f(f(x,x),y) \to f(f(0,f(1,2)),y) \tag{2}$$

Note that $t = f(f(0,1), f(0,f(1,2)))$ is non-$(\mathcal{E}xt_{AC}(R), AC)$-terminating:

$$\begin{array}{l} \underline{f(f(0,1),f(0,f(1,2)))} \sim_A f(0,\underline{f(1,f(0,f(1,2)))}) \sim_A f(0,f(\underline{f(1,0)},f(1,2))) \sim_C \\ f(0,\underline{f(f(0,1),f(1,2))}) \sim_A \underline{f(0,f(0,f(1,f(1,2))))} \sim_A \underline{f(f(0,0),f(1,f(1,2)))} \xrightarrow{\Lambda}_{\mathcal{E}xt_{AC}(R)} \\ f(f(0,f(1,2)),f(1,f(1,2))) \to_{\mathcal{E}xt_{AC}(R),AC} \cdots \end{array}$$

Since $f(0,1)$ and $f(0,f(1,2))$ are in $(\mathcal{E}xt_{AC}(R), AC)$-normal form, we have that $t \in \mathcal{T}_{\infty,R,AC}$. However, $t' = f(f(0,0), f(1,f(1,2)))$, which is AC-equivalent to t (i.e., $t \sim_{AC} t'$), is non-AC-terminating but it is *not* minimal because its strict subterm $f(1,f(1,2)))$ is non-$(\mathcal{E}xt_{AC}(R), AC)$-terminating:

$$\begin{array}{l} \underline{f(1,f(1,2)))} \sim_A \underline{f(f(1,1),2)} \xrightarrow{\Lambda}_{\mathcal{E}xt_{AC}(R)} \underline{f(f(0,f(1,2)),2)} \sim_A f(0,\underline{f(f(1,2),2)}) \\ \sim_A \underline{f(0,f(1,f(2,2)))} \sim_A \underline{f(f(0,1),f(2,2))} \sim_C \underline{f(f(2,2),f(0,1))} \xrightarrow{\Lambda}_{\mathcal{E}xt_{AC}(R)} \\ f(f(0,f(1,2)),f(0,1)) \to_{\mathcal{E}xt_{AC}(R),AC} \cdots \end{array}$$

Example 2 shows that an essential property of minimal terms when considered as part of infinite $(\mathcal{E}xt_E(R), E)$-rewriting sequences for $A\vee C$-theories E gets lost: the application of $(\mathcal{E}xt_E(R), E)$-rewrite steps *at the root* of a minimal term s by means of a rule $l \to r$ (i.e., $s \sim_{AC} \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$) does *not* guarantee that there is a *nonvariable subterm* v of the right-hand side r which is a prefix of the 'next' minimal term in the infinite sequence.

Example 3. Term t in Example 2 can be rewritten at the root *only* by rule (2) of $\mathcal{E}xt_{AC}(R)$. We can apply this rule to t' in Example 2 (for instance) to obtain $s' = \sigma(r) = f(f(0, f(1,2)), f(1, f(1,2)))$ (where $r = f(f(0, f(1,2)), y)$), which is non-$(\mathcal{E}xt_{AC}(R), AC)$-terminating. Note that s' contains a minimal term $u \in \mathcal{T}_{\infty,R,E}$. Since $s'|_2 = f(1, f(1,2))$ is non-$(\mathcal{E}xt_{AC}(R), AC)$-terminating, it follows that s' is *not* minimal. Since $s'|_1 = f(0, f(1,2))$ is $(\mathcal{E}xt_{AC}(R), AC)$-terminating, the only possibility is that u occurs in $s'|_2$. Actually, $s'|_2$ is minimal already; hence, $u = s'|_2$. But note the absence of any nonvariable position $p \in \mathcal{P}os(r)$ in the right-hand side of the considered rule such that $\sigma(r|_p) = u = f(1, f(1,2))$.

This is in sharp contrast with the situation of the DP-approach for ordinary rewriting. Furthermore, it is not difficult to see that for all $t'' \sim_{AC} t$ such $t'' = \sigma'(l)$ for some substitution σ', we have a similar situation. Thus, the problem illustrated here cannot be solved by using a different $\sim_{AC}$ sequence before performing the $\mathcal{E}xt_{AC}(R)$-root-step.

In the following we introduce a new notion of minimality which solves these problems.

3.1 A New Notion of Minimal Non-E-Terminating Terms

The following definition solves the problems discussed above by explicitly requiring that the condition defining minimality is preserved under E-equivalence.

Definition 3 (Stably minimal non-E-terminating term). *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. Let $\mathcal{M}_{\infty,R,E}$ be a set of stably minimal non-E-terminating terms in the following sense: $t \in \mathcal{T}(\Sigma, \mathcal{X})$ belongs to $\mathcal{M}_{\infty,R,E}$ if t is non-E-terminating, and for all $t' \sim_E t$ and every proper subterm s' of t' (i.e., $t' \rhd s'$), s' is $(\mathcal{E}xt_E(R), E)$-terminating.*

We have the following useful characterization of minimality.

Proposition 2 (Characterization of stably minimal terms). *Let $\mathcal{R} = (\Sigma, R, E)$ be a rewrite theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, $t \in \mathcal{M}_{\infty,R,E}$ if and only if $[t]_E \subseteq \mathcal{T}_{\infty,R,E}$. Therefore, $\mathcal{M}_{\infty,R,E} = \{t \in \mathcal{T}(\Sigma, \mathcal{X}) \mid [t]_E \subseteq \mathcal{T}_{\infty,R,E}\}$.*

The problem in Example 2 disappears now: t is *not* minimal according to Definition 3. The following result shows how to *find* stably minimal non-E-terminating terms associated to a given non-E-terminating term. This is essential in our development.

Proposition 3. *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory such that $[t]_E = \{t\}$ for all constant and variable terms t. Let $s \in \mathcal{T}(\Sigma, \mathcal{X})$. If s is non-E-terminating, then there is a subterm t of some $s' \sim_E s$ ($s' \trianglerighteq t$) such that $t \in \mathcal{M}_{\infty,R,E}$.*

Clearly, Proposition 3 holds whenever $\mathcal{R}$ is an $A \vee C$-rewrite theory.

Example 4. Consider the term t in Example 2. Although $t \in \mathcal{T}_{\infty,R,E}$, $t \notin \mathcal{M}_{\infty,R,E}$: the term $t' = f(f(0,0), f(1, f(1,2)))$, which is AC-equivalent to t, contains a subterm $u = f(1, f(1,2))$ which is non-E-terminating. It is not difficult to see that actually $u \in \mathcal{M}_{\infty,R,E}$.

In general, Proposition 3 does *not* hold for arbitrary sets of equations E.

Example 5. Consider the following example [5, Example 13]:

$$R : f(x) \to x \qquad E : f(a) = a$$

Note that $a \in \mathcal{T}_{\infty,R,E}$. However, a is *not* stably minimal because $a \sim_E f(a)$ but $f(a) \notin \mathcal{T}_{\infty,R,E}$. Thus, Proposition 3 does not hold.

Now we provide a more precise result about where we can find stably minimal subterms within a non-E-terminating term for $A \vee C$-rewrite theories $\mathcal{R} = (\Sigma, E, R)$. In the following theorem, given a term s and a symbol f, by an f-subterm t of s (written $s \trianglerighteq_f t$) we mean a subterm t of s such that $t = s|_p$ and for all $q < p$, $root(s|_q) = f$. We also write $s \triangleright_f t$ if $s \trianglerighteq_f t$ and $s \neq t$.

Theorem 2. *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. If s is non-E-terminating, then there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s ($s \trianglerighteq t$) and*

1. *If (1) $A_{root(t)} \notin E_{root(t)}$ or (2) $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then $t \in \mathcal{M}_{\infty,R,E}$.*
2. *If $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and $t \notin \mathcal{M}_{\infty,R,E}$, then there is $s' \sim_E t$ and a strict f-subterm u of s' ($s' \triangleright_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

The following result is just a convenient reformulation of the previous one.

Corollary 1. *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. If s is non-E-terminating, then either there is a subterm $t \in \mathcal{M}_{\infty,R,E}$ of s ($s \trianglerighteq t$), or there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s satisfying that $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and such that there is $s' \sim_E t$ and a strict f-subterm u of s' ($s' \triangleright_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

4 Structure of (Stably) Minimal Infinite $A \vee C$-Rewrite Sequences

Now we analyze $A \vee C$-rewrite sequences starting from stably minimal non-$A \vee C$-terminating terms. First we consider a restricted case.

Proposition 4. *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A \vee C$-rewrite theory. Let $s \in \mathcal{M}_{\infty,R,E}$ be such that $f = root(s)$ and either (1) $A_f \notin E_f$, or (2) $s = f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1)$, $root(s_2) \in \mathcal{C}$. Assume that for all $l \to r \in R$ such that $root(l) = f$ and all subterms v of r ($r \trianglerighteq v$) such that $v = g(v_1, v_2)$ for some associative symbol g, we have that $root(v_1), root(v_2) \notin \mathcal{X} \cup \{g\}$. Then, there exist $l \to r \in R$, a substitution σ and terms $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and $u \in \mathcal{M}_{\infty,R,E}$ such that*

$$s \xrightarrow[\mathcal{E}xt_E(R),E]{>\Lambda}{}^{*} t \sim_E \sigma(l) \xrightarrow{\Lambda}_R \sigma(r) \trianglerighteq u$$

and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.

Unfortunately, stable minimality of (arbitrary) non-E-terminating terms s for $A \vee C$-theories E is not preserved under inner $(\mathcal{E}xt_E(R), E)$-rewritings.

Example 6. Term $u = f(f(1,1),2)$ in Example 2 is stably minimal: $u \in \mathcal{M}_{\infty,R,E}$. We have that $f(\underline{f(1,1)},2) \xrightarrow{>\Lambda}_R f(f(0,f(1,2)),2)$. Note that $f(f(0,f(1,2)),2) \notin \mathcal{M}_{\infty,R,E}$: we have $\underline{f(f(0,f(1,2)),2)} \sim_A f(0,\underline{f(f(1,2),2)}) \sim_A f(0,f(1,f(2,2)))$ where $f(0,f(1,\underline{f(2,2)}))$ contains a subterm $f(1,f(2,2))$ which is non-$(\mathcal{E}xt_E(R), E)$-terminating.

In the following, we show how to avoid this problem. We define *deep reduction* as a restriction $\xrightarrow{>1,2}_{\mathcal{E}xt_E(R),E}$ of inner $(\mathcal{E}xt_E(R), E)$-rewriting which restricts reductions on terms like u above. We will show that deep reduction preserves stable minimality of non-E-terminating terms for $A \vee C$-rewrite theories $\mathcal{R} = (\Sigma, E, R)$.

Definition 4 (Deep reduction). *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. Given $t \in \mathcal{T}(\Sigma, \mathcal{X})$, $t \xrightarrow{>1,2}_{\mathcal{E}xt_E(R),E} s$ if $t \xrightarrow{q}_{\mathcal{E}xt_E(R),E} s$ for some position $q \in \mathcal{P}os(t)$ such that $q > p$ for $p \in \{1,2\}$ if $t = \sigma(u)$ for some $u = v \in E$ or $v = u \in E$ and $u|_p \notin \mathcal{X}$; otherwise, $q > \Lambda$.*

Obviously, $\xrightarrow{>1,2}_{\mathcal{E}xt_E(R),E} \subseteq \xrightarrow{>\Lambda}_{\mathcal{E}xt_E(R),E}$. The following proposition shows that *deep reduction* preserves stable minimality.

Proposition 5. *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory and $t \in \mathcal{M}_{\infty,R,E}$. If $t \xrightarrow{>1,2}{}^*_{\mathcal{E}xt(\mathcal{R}),E} s$ and s is non-E-terminating, then $s \in \mathcal{M}_{\infty,R,E}$.*

As a consequence, the following theorem establishes the desired property for stable minimal non-$A \vee C$-terminating terms.

Theorem 3. *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. For all $s \in \mathcal{M}_{\infty,R,E}$, there exist $l \to r \in \mathcal{E}xt_E(R)$ and a substitution σ such that*

$$s \, (\sim_E \circ \xrightarrow{>1,2}_{\mathcal{E}xt_E(R),E})^* t \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$$

and there is a nonvariable subterm v of r ($r \trianglerighteq v$), such that either

1. *$v = f(v_1, v_2)$ for some associative symbol f, $root(v_1) \in \mathcal{X} \cup \{f\}$ or $root(v_2) \in \mathcal{X} \cup \{f\}$, $root(\sigma(v_1)) = f$ or $root(\sigma(v_2)) = f$, $\sigma(v) \in \mathcal{T}_{\infty,R,E}$ and there is a term $t' \sim_E \sigma(v)$ containing a strict f-subterm $u = f(u_1, u_2)$ ($t' \rhd_f u$) such that $u \in \mathcal{M}_{\infty,R,E}$, or*
2. *$\sigma(v) \in \mathcal{M}_{\infty,R,E}$ otherwise.*

Example 2 shows that Theorem 3 does not hold for Giesl and Kapur's minimal terms $s \in \mathcal{T}_{\infty,R,E}$.

5 $A \vee C$-Dependency Pairs and Chains

Propositions 3 and 4 together with Theorem 3 are the basis for our definition of *$A \vee C$-Dependency Pairs* and the corresponding *chains*. Together, they show

that given an $A \vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$, every non-$E$-terminating term s has an associated infinite $(\mathcal{E}xt_E(R), E)$-rewrite sequence starting from a stably minimal subterm $t \in \mathcal{M}_{\infty,R,E}$. Such a sequence proceeds as described in Proposition 4 and Theorem 3, depending on the shape of t.

This process is abstracted in the following definition of $A \vee C$-dependency pairs (Definition 5) and in the definition of chain below (Definition 6).

Given a signature Σ and $f \in \Sigma$, we let $f^\sharp$ denote a fresh new symbol (often called *tuple* symbol or DP-symbol) associated to a symbol f [1]. Let $\Sigma^\sharp$ be the set of tuple symbols associated to symbols in Σ. As usual, for $t = f(t_1, \ldots, t_k) \in \mathcal{T}(\Sigma, \mathcal{X})$, we write $t^\sharp$ to denote the *marked* term $f^\sharp(t_1, \ldots, t_k)$ (written sometimes $F(t_1, \ldots, t_k)$). Given a set of rules R and a symbol $f \in \Sigma$, we let $R_f = \{l \to r \in R \mid root(l) = f\}$. Given a set of rules R, the set $\mathsf{DP}(R)$ of dependency pairs associated to R is [1]: $\mathsf{DP}(R) = \{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq s, root(s) \in \mathcal{D}\}$.

Definition 5 ($A \vee C$-Dependency Pairs). *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A \vee C$-rewrite theory. Then, $\mathsf{DP}_E(R) = \mathsf{DP}(\mathcal{E}xt_E(R))$ is the set of $A \vee C$-dependency pairs ($A \vee C$-DPs) of $\mathcal{R}$.*

In general, the set of $A \vee C$-DPs which is obtained from Definition 5 is a subset of those which are obtained by particularizing Giesl and Kapur's definitions to the $A \vee C$ case [5].

Example 7. Consider the AC-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 2. The set $DP_E(R)$ consists of the following pairs:

$$F(x, x) \to F(0, f(1, 2)) \tag{3}$$
$$F(x, x) \to F(1, 2) \tag{4}$$
$$F(f(x, x), y) \to F(f(0, f(1, 2)), y) \tag{5}$$
$$F(f(x, x), y) \to F(0, f(1, 2)) \tag{6}$$
$$F(f(x, x), y) \to F(1, 2) \tag{7}$$

5.1 Chains of $A \vee C$-DPs

An essential property of the dependency pair method is that it provides a *characterization* of termination of TRSs $\mathcal{R}$ as the absence of infinite (minimal) *chains of dependency pairs* [1,6]. If we want to prove the same for $A \vee C$-rewrite theories, we have to introduce a suitable notion of chain which can be used with $A \vee C$-DPs. As in the DP-framework, where the origin of *pairs* does not matter, we should rather think of another rewrite theory $\mathcal{P} = (\Gamma, F, P)$ which is used together with $\mathcal{R}$ to build the chains. According to the usual terminology [6], we often call *pairs* to the rules $u \to v \in P$.

Definition 6 (Chain of pairs - Minimal chain). *Let $\mathcal{P} = (\Gamma, F, P)$ and $\mathcal{R} = (\Sigma, E, R)$ be rewrite theories, and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. An (F, P, E, R, S)-chain is a finite or infinite sequence of pairs $u_i \to v_i \in P$, together with substitutions σ and θ_i satisfying that, for all $i \geq 1$:*

1. *If* $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ *satisfies* $\sigma(v_i) = \theta_i(u'_i)$ *for some* $u'_i = v'_i \in F$ *or* $v'_i = u'_i \in F$ *such that* $u'_i = f_i(u'_{i1}, u'_{i2})$ *satisfies* $u'_{i1} \notin \mathcal{X}$ *or* $u'_{i2} \notin \mathcal{X}$*, then*

$$\sigma(v_i) \sim_F \circ \xrightarrow{\Lambda}{}^{+}_{\mathcal{S}_{f_i}} t_i \, (\sim_F \circ \xrightarrow{>1,2}_{\mathcal{E}xt_E(R),E})^* \circ \sim_F \sigma(u_{i+1})$$

2. *and* $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \sim_F \sigma(u_{i+1})$*, otherwise.*

An (F, P, E, R, S)*-chain is called* minimal *if for all* $i \geq 1$*, and* $t'_i \in [t_i]_F$*,* t'_i *is* $(\mathcal{E}xt_E(R), E)$*-terminating.*

As usual, in Definition 6 we assume that different occurrences of dependency pairs do not share any variable (renaming substitutions are used if necessary).

This more abstract notion of chain can be particularized to be used with $A\vee C$-DPs, by just taking

1. $P = \mathsf{DP}_E(R)$,
2. $F = E \cup E^\sharp$, where $E^\sharp = \{s^\sharp = t^\sharp \mid s = t \in E\}$, and
3. $\mathcal{S} = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$.

Theorem 4 (Characterization of $A\vee C$-termination). *Let* $\mathcal{R} = (\Sigma, E, R)$ *be an* $A\vee C$*-rewrite theory. Let* $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ *be a TRS such that* $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$*. Then,* $\mathcal{R}$ *is* $(\mathcal{E}xt_E(R), E)$*-terminating if and only if there is no infinite minimal* $(E^\sharp \cup E, \mathsf{DP}_E(R), E, R, S)$*-chain.*

6 An $A\vee C$-Dependency Pair Framework

In the following, we adapt Giesl et al. DP-framework to provide a suitable framework for mechanizing proofs of $A\vee C$-termination using $A\vee C$-DPs.

Definition 7 ($A\vee C$ problem). *An $A\vee C$ problem τ is a tuple $\tau = (F, P, E, R, S)$, where $\mathcal{R} = (\Sigma, E, R)$ is an $A\vee C$-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. An $A\vee C$ problem is finite if there is no infinite minimal (F, P, E, R, S)-chain. An $A\vee C$ problem τ is infinite if $\mathcal{R}$ is non-$A\vee C$-terminating or there is an infinite minimal (F, P, E, R, S)-chain.*

The following definition adapts the notion of *processor* [6] to prove termination of $A\vee C$-rewrite theories.

Definition 8 ($A\vee C$ processor). *An $A\vee C$ processor* Proc *is a mapping from $A\vee C$ problems into sets of $A\vee C$ problems. Alternatively, it can also return* "no"*. An $A\vee C$ processor* Proc *is*

- sound *if for all $A\vee C$ problems τ, τ is finite whenever* $\mathsf{Proc}(\tau) \neq \mathsf{no}$ *and* $\forall \tau' \in \mathsf{Proc}(\tau)$*, τ' is finite.*
- complete *if for all $A\vee C$ problems τ, τ is infinite whenever* $\mathsf{Proc}(\tau) = \mathsf{no}$ *or* $\exists \tau' \in \mathsf{Proc}(\tau)$ *such that τ' is infinite.*

Similar to [6] for the DP-framework, we construct a tree whose nodes are labeled with $A \vee C$ problems or "yes" or "no", and whose root is labeled with $(E^\sharp \cup E, \mathsf{DP}_E(R), E, R, S)$. Now we have the following result which adapts [6, Corollary 5] to $A \vee C$-rewrite theories.

Theorem 5 ($A \vee C$-DP framework). *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-theory. We construct a tree whose nodes are labeled with $A \vee C$ problems or* "yes" *or* "no"*, and whose root is labeled with $(E^\sharp \cup E, \mathsf{DP}_E(R), E, R, S)$, where $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. For every inner node labeled with τ, there is a sound processor* Proc *satisfying one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child, labeled with* "no"*.*
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child, labeled with* "yes"*.*
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$*, and the children of the node are labeled with the $A \vee C$ problems in* $\mathsf{Proc}(\tau)$*.*

If all leaves of the tree are labeled with "yes"*, then $\mathcal{R}$ is E-terminating. Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is not E-terminating.*

6.1 $A \vee C$-Dependency Graph

$A \vee C$ problems focus our attention on the analysis of *infinite minimal chains*. Our aim here is obtaining a notion of graph which is able to represent all infinite *minimal* chains of pairs as given in Definition 6.

Definition 9 ($A \vee C$-Graph of Pairs). *Let $\mathcal{R} = (\Sigma, E, R)$ and $\mathcal{P} = (\Gamma, F, P)$ be rewrite theories and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. The* $A \vee C$-graph *associated to them (denoted $\mathsf{G}(F, P, E, R, S)$) has P as the set of nodes. There is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $u \to v, u' \to v'$ is an (F, P, E, R, S)-chain.*

In termination proofs, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [8]. A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle which is not contained in any other cycle. In the following result, given two sets of rules S and Q, we let S_Q be the least subset of S satisfying that whenever there is a rule $u \to v \in Q$, such that v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then $S_f \subseteq S_Q$.

Theorem 6 (SCC processor). *Let $\mathcal{R} = (\Sigma, E, R)$ and $\mathcal{P} = (\Gamma, F, P)$ be rewrite theories and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. Then, the processor Proc_{SCC} given by*

$$\mathsf{Proc}_{SCC}(F, P, E, R, S) = \{(F, Q, E, R, S_Q) \mid Q \text{ are the pairs of an SCC in } \mathsf{G}(F, P, E, R, S)\}$$

is sound and complete.

As a consequence, we can *separately* work with the strongly connected components of $\mathsf{G}(F, P, E, R, S)$, disregarding other parts of the graph. Now we can use these notions to introduce the $A \vee C$-dependency graph, i.e., the $A \vee C$-graph whose nodes are the $A \vee C$-DPs instead of an arbitrary set of pairs.

Definition 10 ($A \vee C$-Dependency Graph). *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. The $A \vee C$-*Dependency Graph *associated to $\mathcal{R}$ is: $\mathsf{DG}(\mathcal{R}) = \mathsf{G}(E^\sharp \cup E, \mathsf{DP}_E(R), E, R, S)$.*

6.2 Estimating the $A \vee C$-Dependency Graph

As in standard rewriting, the $A \vee C$-dependency graph of an $A \vee C$-rewrite theory is in general *not* computable. So, we need to use some approximation of it. For any term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ let $\textsc{Cap}(t)$ result from replacing all proper subterms rooted by a defined symbol by fresh variables and let $\textsc{Ren}(t)$ which *independently* renames all *occurrences* of variables in t by using new fresh variables [1].

As usual, we do not have to talk about *mgu* when dealing with rewriting modulo equations. Instead, it is used the notion of complete set of E-unifiers. However, although in theory, all these unifiers have to be considered, for our results of reachability it is enough to check the existence of one.

Proposition 6. *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $u, t \in \mathcal{T}(\Sigma, \mathcal{X})$ be such that $\mathcal{V}ar(u) \cap \mathcal{V}ar(t) = \varnothing$ and θ, θ' be substitutions. If $\theta(t) \to^*_{\mathcal{E}xt_E(R),E} \circ \sim_E \theta'(u)$, then $\textsc{Ren}(\textsc{Cap}(t))$ and u E-unify.*

Now, we are ready to provide a correct estimation of our graph of pairs. Correctness of our definition relies on Proposition 6.

Definition 11 (Estimated $A \vee C$-Graph of Pairs). *Let $\mathcal{R} = (\Sigma, E, R)$ and $\mathcal{P} = (\Gamma, F, P)$ be rewrite theories and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. The* estimated $A \vee C$-graph *associated to them (denoted $\mathsf{EG}(F, P, E, R, S)$) has P as the set of nodes and arcs which connect them as follows:*

1. *If v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $root(u') = f$.*
2. *Otherwise, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $\textsc{Ren}(\textsc{Cap}(v))$ and u' E-unify.*

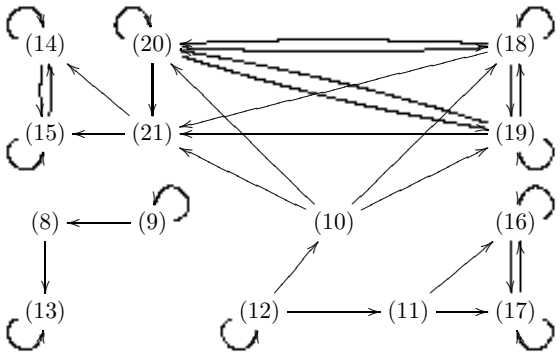
According to Definition 9, we would have the corresponding one for the *estimated* $A \vee C$-DG: $\mathsf{EDG}(\mathcal{R}) = \mathsf{EG}(E^\sharp \cup E, \mathsf{DP}_E(R), E, R, S)$, where

$$S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

Example 8. For the $A \vee C$-rewrite theory in Figure 1, the set $\mathsf{DP}_E(R)$ is[1]:

$$LIST2SET(cons(N,L)) \to UNION(N, list2set(L)) \quad (8)$$
$$LIST2SET(cons(N,L)) \to LIST2SET(L) \quad (9)$$
$$IN(N, union(M,S)) \to EQ(N,M) \quad (10)$$
$$IN(N, union(M,S)) \to OR(eq(N,M), in(N,S)) \quad (11)$$
$$IN(N, union(M,S)) \to IN(N,S) \quad (12)$$
$$UNION(union(N,N),Z) \to UNION(N,Z) \quad (13)$$
$$AND(and(true,B),Z) \to AND(B,Z) \quad (14)$$
$$AND(and(false,B),Z) \to AND(false,Z) \quad (15)$$
$$OR(or(true,B),Z) \to OR(true,Z) \quad (16)$$
$$OR(or(false,B),Z) \to OR(B,Z) \quad (17)$$
$$EQ(s(N),s(M)) \to EQ(N,M) \quad (18)$$
$$EQ(cons(N,L),cons(M,L')) \to EQ(N,M) \quad (19)$$
$$EQ(cons(N,L),cons(M,L')) \to EQ(L,L') \quad (20)$$
$$EQ(cons(N,L),cons(M,L')) \to AND(eq(N,M),eq(L,L')) \quad (21)$$

The (estimated) $A \vee C$-DG is:



By Theorem 6 we transform the $A \vee C$ problem $(E \cup E^\sharp, \mathsf{DP}(R), E, R, S)$ into a set $\mathsf{Proc}_{SCC}(E \cup E^\sharp, \mathsf{DP}(R), E, R, S)$ given by

$$\{(E \cup E^\sharp, \{(9)\}, E, R, \varnothing), (E \cup E^\sharp, \{(12)\}, E, R, \varnothing), (E \cup E^\sharp, \{(13)\}, E, R, S_{union}),$$

$$(E \cup E^\sharp, \{(14),(15)\}, E, R, S_{and}), (E \cup E^\sharp, \{(16),(17)\}, E, R, S_{or}), (E \cup E^\sharp, \{(18),(19),(20)\}, E, R, \varnothing)\}$$

which contains six new (but simpler) $A \vee C$ problems.

6.3 Use of Reduction Pairs

A reduction pair $(\gtrsim, \sqsupset)$ consists of a stable and monotonic quasi-ordering $\gtrsim$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. In the dependency pair framework reduction pairs are used to obtain *smaller*

[1] We have introduced new 'prefix' symbols *eq*, *cons* and *union* instead of the original 'infix' ones $\_==\_$, $\_;\_$, $\_,\_$.

sets of pairs $\mathcal{P}' \subseteq \mathcal{P}$ by removing the *strict* pairs, i.e., those pairs $u \to v \in \mathcal{P}$ such that $u \sqsupset v$. Stability is required both for $\gtrsim$ and $\sqsupset$ because, although we only check the left- and right-hand sides of the rewrite rules $l \to r$ (with $\gtrsim$) and pairs $u \to v$ (with $\gtrsim$ or $\sqsupset$), the chains of pairs involve *instances* $\sigma(l)$, $\sigma(r)$, $\sigma(u)$, and $\sigma(v)$ of rules and pairs and we aim at concluding $\sigma(l) \gtrsim \sigma(r)$, and $\sigma(u) \gtrsim \sigma(v)$ or $\sigma(u) \sqsupset \sigma(v)$, respectively. Monotonicity is required for $\gtrsim$ to deal with the application of rules $l \to r$ to an arbitrary depth in terms. Since the pairs are 'applied' only at the root level, no monotonicity is required for $\sqsupset$ (but, for this reason, we cannot compare the rules in $\mathcal{R}$ using $\sqsupset$). Dealing with associative-commutative axioms, we will compare them with the equivalence relation defined by the stable, reflexive, transitive, and symmetric equivalence $\sim$ induced by $\gtrsim$, i.e., $\sim \,= \gtrsim \cap \lesssim$, since we need to impose compatibility with the equational theories E and F. The following theorem formalizes a generic processor to remove pairs from $\mathcal{P}$ by using reduction pairs.

Theorem 7 (Reduction pair processor). *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. Let $(\gtrsim, \sqsupset)$ be a reduction pair such that*

1. $R \subseteq \gtrsim$,
2. $P \cup S \subseteq \gtrsim \cup \sqsupset$, *and*
3. $E \cup F \subseteq \sim$.

Let $P_{\sqsupset} = \{u \to v \in P \mid u \sqsupset v\}$. Then, the processor Proc_{RP} *given by*

$$\mathsf{Proc}_{RP}(F, P, E, R, S) = \begin{cases} \{(F, P - P_{\sqsupset}, E, R, S)\} & \text{if (1), (2), and (3) hold} \\ \{(F, P, E, R, S)\} & \text{otherwise} \end{cases}$$

is sound and complete.

7 Related Work and Conclusions

As remarked in the introduction, this is not the first work which tries to use dependency pairs for proving termination of rewriting modulo an equational theory, see [5,10,11,9,13,14]. Our work, however, is, as far as the authors know, the first one which provides a correct notion of minimal non-terminating term for an $A \vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ which can be used to provide a suitable definition of minimal chain of dependency pairs which can be used to characterize $A \vee C$-termination (Theorem 4). In order to substantiate this claim, consider the AC-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 2 again. The $A \vee C$-DPs for $\mathcal{R}$ are enumerated in Example 7. Such dependency pairs coincide with the ones which would be computed by, e.g., [5,10,11]. Remember that t in Example 2 is *minimal* in Giesl and Kapur's sense (Definition 2). We should, then, be able to find an infinite *minimal* chain of DPs starting from $t^\sharp$. According to [5,10,11], 'minimal' means that $\sigma(v_i)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all pairs $u_i \to v_i \in \mathsf{DP}_E(R)$ in the chain of dependency pairs induced by the substitution σ. However, this is *not* possible: the marked version $t^\sharp$ of t is $F(f(0,1), f(0, f(1,2)))$, which is an

$(\mathcal{E}xt_E(R), E)$-terminating term. After some $E^\sharp \cup E$-equivalence steps we would be able to apply one of the rules in $\mathsf{DP}_E(R)$. Note, however, that *no rule* $u \to v \in \mathsf{DP}_E(R)$ except (5) has a right-hand side v which can be rewritten (after instantiation into $\sigma(v)$) into an instance $\sigma(u')$ of the left-hand side u' of any other pair in $\mathsf{DP}_E(R)$ by means of $(\mathcal{E}xt_E(R), E^\sharp \cup E)$-rewriting steps. This means that only the dependency pair (5) could be used in any infinite minimal chain of dependency pairs starting from $t^\sharp$. But such a chain would start as follows:

$$F(f(0,1), f(0, f(1,2))) \sim_{E^\sharp \cup E} F(f(0,0), f(1, f(1,2))) \to_{(5)} F(f(0, f(1,2)), f(1, f(1,2)))$$

where $F(f(0, f(1,2)), f(1, f(1,2)))$ contains a subterm $f(1, f(1,2))$ which, as showed in Example 2, is non-$(\mathcal{E}xt_E(R), E)$-terminating. Therefore, this chain of dependency pairs is *not* minimal. We conclude that, according to the notion of minimal chain in the aforementioned papers, *there is no minimal chain of pairs starting from* $t^\sharp$. This means that no *sound* approach to proving AC-termination on the basis of such notion of minimal chain is possible. In this paper we have introduced the notion of *stably minimal term* (Definition 3) which overcomes these problems (Proposition 4 and Theorem 3) and leads to an appropriate characterization of $A \vee C$-termination as the absence of infinite minimal chains of $A \vee C$-DPs (Definitions 5 and 6, and Theorem 4).

Furthermore, we note that [10,11] deal with *AC-rewrite theories* only, and that [5], which considers more general rewrite theories E including $A \vee C$-theories do not cover our work in a second respect: when purely associative theories are considered (i.e., rewrite theories $\mathcal{R} = (\Sigma, E, R)$ such that $E_f \subseteq \{A_f\}$ for all $f \in \Sigma$), then Giesl and Kapur's technique requires the computation of *instances* of the rules in $\mathcal{E}xt_E(R)$ for which the computation of *all* the E-unifiers $uni_E(v, l)$ of v and l for the rules $l \to r$ in $\mathcal{E}xt_E(R)$ and equations $u = v \in E$ or $v = u \in E$ is required. It is well-known, however, that the E-unification problem for associative theories E is *infinitary*, which means that $uni_E(v, l)$ is not guaranteed to be finite, in general. In sharp contrast, we do not have to do that for dealing with purely associative rewrite theories $\mathcal{R}$.

Our second main (and novel) contribution is the formalization of an $A \vee C$-dependency pair framework (Definitions 7 and 8) which, on the basis of the previously developed theory, can be used to develop automatic tools for proving termination of $A \vee C$-rewrite theories (Theorem 5). Two important processors have been adapted as well: the SCC processor (Theorem 6) and the reduction pair processor (Theorem 7).

Much work remains ahead both in terms of further developing the new $A \vee C$-dependency pair framework and in tool support. Appropriate reduction orderings which are well-suited for being used in the reduction pair processor should be investigated. It would also be very useful to explore how the requirements on E can be relaxed to handle even more general sets of axioms. Regarding tool support for the method we have presented, we plan to integrate it within the tool MU-TERM [2]. In this way, our termination technique modulo combinations of associative and commutative axioms will become applicable to an even wider range of rewrite theories, that can be transformed into $A \vee C$-theories by non-termination-preserving transformations [3,4,12].

References

1. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236(1-2), 133–178 (2000)
2. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. Electronic Notes in Theoretical Computer Science 188, 105–115 (2007)
3. Durán, F., Lucas, S., Meseguer, J.: Termination Modulo Combinations of Equational Theories. In: Ghilardi, S., Sebastiani, R. (eds.) FroCoS 2009. LNCS (LNAI), vol. 5749, pp. 246–262. Springer, Heidelberg (2009)
4. Durán, F., Lucas, S., Marché, C., Meseguer, J., Urbain, X.: Proving Operational Termination of Membership Equational Programs. Higher-Order and Symbolic Computation 21(1-2), 59–88 (2008)
5. Giesl, J., Kapur, D.: Dependency Pairs for Equational Rewriting. In: Middeldorp, A. (ed.) RTA 2001. LNCS, vol. 2051, pp. 93–108. Springer, Heidelberg (2001)
6. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. Journal of Automatic Reasoning 37(3), 155–203 (2006)
7. Hirokawa, N., Middeldorp, A.: Dependency Pairs Revisited. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 249–268. Springer, Heidelberg (2004)
8. Hirokawa, N., Middeldorp, A.: Automating the Dependency Pair Method. Information and Computation 199, 172–199 (2005)
9. Kusakari, K.: Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems. PhD. Thesis, School of Information Science, JAIST (2000)
10. Kusakari, K., Nakamura, M., Toyama, Y.: Elimination Transformations for Associative-Commutative Rewriting Systems. Journal of Automated Reasoning 37, 205–229 (2006)
11. Kusakari, K., Toyama, Y.: On Proving AC-Termination by AC-Dependency Pairs. IEICE Transactions on Information and Systems, E84-D, 604–612 (2001)
12. Lucas, S., Meseguer, J.: Operational Termination of Membership Equational Programs: the Order-Sorted Way. Electronic Notes in Theoretical Computer Science 238(3), 207–225 (2009)
13. Marché, C., Urbain, X.: Termination of associative-commutative rewriting by dependency pairs. In: Nipkow, T. (ed.) RTA 1998. LNCS, vol. 1379, pp. 241–255. Springer, Heidelberg (1998)
14. Marché, C., Urbain, X.: Modular and incremental proofs of AC-termination. Journal of Symbolic Computation 38, 873–897 (2004)
15. Peterson, G.E., Stickel, M.E.: Complete Sets of Reductions for Some Equational Theories. Journal of the ACM 28(2), 233–264 (1981)

18.9 Proving Termination Properties with MU-TERM

9. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. **Proving Termination Properties with** MU-TERM. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology And Software Technology, AMAST'10*, volume 6486 of *Lecture Notes in Computer Science*, pages 201–208. Springer-Verlag, 2010.

Proving Termination Properties with MU-TERM*

Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas, and Rafael Navarro-Marset

ELP group, DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, E-46022 Valencia, Spain

Abstract. MU-TERM is a tool which can be used to verify a number of termination properties of (variants of) Term Rewriting Systems (TRSs): termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and termination of rewriting modulo specific axioms. Such termination properties are essential to prove termination of programs in sophisticated rewriting-based programming languages. Specific methods have been developed and implemented in MU-TERM in order to efficiently deal with most of them. In this paper, we report on these new features of the tool.

1 Introduction

Handling typical programming language features such as sort/types and subtypes, evaluation modes (eager/lazy), programmable strategies for controlling the execution, rewriting modulo axioms and so on is outside the scope of many termination tools. However, such features can be very important to determine the termination behavior of programs. For instance, in Figure 1 we show a Maude [10] program encoding an *order-sorted* TRS which is terminating when the sorting information is taken into account but which is *nonterminating* as a TRS (i.e., disregarding sort information) [18]. The predicate `is-even` tests whether an integer number is even. When disregarding any information about sorts, the program `EVEN` is not terminating due to the last rule for `is-even`, which specifies a recursive call to `is-even`. However, when sorts are considered and the hierarchy among them is taken into account, such recursive call is not longer possible due to the need of binding variable `Y` of sort `NzNeg` to an expression `opposite(Y)` of sort `NzPos`, which is not possible in the (sub)sort hierarchy given by `EVEN`.

The notions coming from the already quite mature theory of termination of TRSs (orderings, reduction pairs, dependency pairs, semantic path orderings, etc.) provide a basic collection of abstractions for treating termination problems. For real programming languages, though, having appropriate adaptations, methods, and techniques for specific termination problems is essential. Giving support to *multiple* extensions of such *classical* termination notions is one of the main goals for developing a new version of our tool, MU-TERM 5.0:

`http://zenon.dsic.upv.es/muterm`

* Partially supported by EU (FEDER) and MICINN grant TIN 2007-68093-C02-02.

M. Johnson and D. Pavlovic (Eds.): AMAST 2010, LNCS 6486, pp. 201–208, 2011.
© Springer-Verlag Berlin Heidelberg 2011

```
fmod EVEN is
  sorts Zero NzNeg Neg NzPos Pos Int Bool .
  subsorts Zero < Neg < Int .              subsorts NzNeg < Neg .
  subsorts Zero < Pos < Int .              subsorts NzPos < Pos .
  op 0 : -> Zero .                         op is-even : Int -> Bool .
  op s : Pos -> NzPos .                    op is-even : NzPos -> Bool .
  op p : Neg -> NzNeg .                    op is-even : NzNeg -> Bool .
  ops true false : -> Bool .               op opposite : NzNeg -> NzPos .
  var X : Pos .                            var Y : NzNeg .
  eq opposite(p(0)) = s(0) .
  eq opposite(p(Y)) = s(opposite(Y)) .
  eq is-even(0) = true .
  eq is-even(s(0)) = false .
  eq is-even(s(s(X))) = is-even(X) .
  eq is-even(Y) = is-even(opposite(Y)) .
endfm
```

Fig. 1. Maude program

MU-TERM [23,2] was originally designed to prove termination of *Context-Sensitive Rewriting* (CSR, [21]), where reductions are allowed only for specific arguments $\mu(f) \subseteq \{1, \ldots, k\}$ of the k-ary function symbols f in the TRS. In this paper we report on the new features included in MU-TERM 5.0, not only to improve its ability to prove termination of CSR but also to verify a number of *other* termination properties of (variants of) TRSs.

In contrast to *transformational* approaches which translate termination problems into a classical termination problem for TRSs, we have developed specific techniques to deal with termination of CSR, innermost CSR, order-sorted rewriting and rewriting modulo specific axioms (associative or commutative) by using dependency pairs (DPs, [7]). Our benchmarks show that direct methods lead to simpler, faster and more successful proofs. Moreover, MU-TERM 5.0 has been rewritten to embrace the dependency pair framework [17], a recent formulation of the dependency pair approach which is specially well-suited for mechanizing proofs of termination.

2 Structure and Functionality of MU-TERM 5.0

MU-TERM 5.0 consists of 47 Haskell modules with more than 19000 lines of code. A web-based interface and compiled versions in several platforms are available at the MU-TERM 5.0 web site. In the following, we describe its new functionalities.

2.1 Proving Termination of Context-Sensitive Rewriting

As in the unrestricted case [7], the *context-sensitive dependency pairs* (CSDPs, [3]) are intended to capture all possible function calls in infinite μ-rewrite sequences. In [2], even though our quite 'immature' CSDP approach was one of our major assets, MU-TERM still used *transformations* [15,25] and the *context-sensitive recursive path ordering* (CSRPO, [9]) in many termination proofs.

Since the developments in [2], many improvements and refinements have been made when dealing with termination proofs of CSR. The most important one has been the development of the *context-sensitive dependency pair framework* (CSDP framework, [3,20]), for mechanizing proofs of termination of CSR. The central notion regarding termination proofs is that of *CS problem*; regarding mechanization of the proofs is that of *CS processor*. Most processors in the standard DP-framework [17] have been adapted to CSR and many specific ones have been developed (see [3,20]). Furthermore, on the basis of the results in [28] we have implemented specific processors to prove the infiniteness of CS problems. Therefore, MU-TERM 5.0 is the first version of MU-TERM which is also able to disprove termination of CSR. In the following table, we compare the performance of MU-TERM 5.0 and the last reported version of the tool (MU-TERM 4.3 [2]) regarding its ability to prove termination of CSR over the context-sensitive category of the *Termination Problem Data Base*[1] (TPDB) which contains 109 examples[2]. The results show the power of the new CSDP framework in MU-TERM 5.0, not only by solving more examples in less time, but also disregarding the need of using transformations or CSRPO for solving them.

Table 1. MU-TERM 4.3 compared to MU-TERM 5.0 in proving termination of CSR

Termination Tool	Total	Yes	No	CSDPs	CSRPO	Transf.	Average (sec)
MU-TERM **5.0**	99/109	95	4	99	0	0	0.95s
MU-TERM **4.3**	64/109	64	0	54	7	3	3.44s

2.2 Proving Termination of Innermost CSR

Termination of *innermost* CSR (i.e., the variant of CSR where only the deepest μ-replacing redexes are contracted) has been proved useful for proving termination of programs in *eager* programming languages like Maude and OBJ* which permit to control the program execution by means of context-sensitive annotations. Techniques for proving termination of innermost CSR were first investigated in [14,22]. In these papers, though, the original CS-TRS $(\mathcal{R}, \mu)$ is *transformed* into a TRS whose *innermost* termination implies the innermost termination for $(\mathcal{R}, \mu)$. In [4], the *dependency pair method* [7] has been adapted to deal with termination proofs of innermost CSR. This is the first proposal of a *direct* method for proving termination of innermost CSR and MU-TERM was the first termination tool able to deal with it. Our experimental evaluation shows that the use of *innermost context-sensitive dependency pairs* (ICSDPs) highly improves over the performance of transformational methods for proving termination of innermost CSR: innermost termination of 95 of the 109 considered CS-TRSs could be proved by using ICSDPs; in contrast, only 60 of the 109 could be proved by using (a combination of) transformations and then using AProVE [16] for proving the innermost

[1] See `http://termination-portal.org/wiki/TPDB`
[2] We have used version 7.0.2 of the TPDB.

termination of the obtained TRS. Another important aspect of innermost CSR is its use for proving termination of CSR as part of the CSDP framework [1]. Under some conditions, termination of CSR and termination of innermost CSR coincide [14,19]. We then switch from termination of CSR to termination of innermost CSR, for which we can apply the existing processors more successfully (see Section 2.6). Actually, we proceed like that in $30-50\%$ of the CSR termination problems which are proved by MU-TERM 5.0 (depending on the particular benchmarks).

2.3 Proving Termination of Order-Sorted Rewriting

In *order-sorted rewriting*, sort information is taken into account to specify the kind of terms that function symbols can take as arguments. Recently, the *order-sorted dependency pairs* have been introduced and proved useful for proving termination of order-sorted TRSs [26]. As a remarkable difference w.r.t. the standard approach, we can mention the notion of *applicable rules* which are those rules which can eventually be used to rewrite terms of a given sort. Another important point is the use of order-sorted matching and unification. To our knowledge, MU-TERM 5.0 is the only tool which implements specific methods for proving termination of OS-TRSs[3]. Our benchmarks over the examples in the literature (there is no order-sorted category in the TPDB yet) show that the new techniques perform quite well. For instance, we can prove termination of the OS-TRS `EVEN` in Figure 1 automatically.

2.4 Proving Termination of $A \vee C$-Rewriting

Recently, we have developed a suitable dependency pair framework for proving termination of $A \vee C$-rewrite theories [5]. An $A \vee C$-rewrite theory is a tuple $\mathcal{R} = (\Sigma, E, R)$ where E is a set containing associative or commutative axioms associated to function symbols of the signature Σ. We have implemented the techniques described in [5] in MU-TERM. Even with only a few processors implemented, MU-TERM behaves well in the equational category of the TPDB, solving 39 examples out of 71. Obviously, we plan to investigate and implement more processors in this field. This is not the first attempt to prove termination of rewriting modulo axioms: C*i*ME [11] is able to prove AC-termination of TRSs, and AProVE is able to deal with termination of rewriting modulo equations satisfying some restrictions.

2.5 Use of Rational Polynomials and Matrix Interpretations

Proofs of termination with MU-TERM 5.0 heavily rely on the generation of polynomial orderings using polynomial interpretations with rational coefficients [24]. In this sense, recent improvements which are new with respect to the previous

[3] The Maude Termination Tool [12] implements a number of *transformations* from OS-TRSs into TRSs which can also be used for this purpose.

versions of MU-TERM reported in [2,23] are the use of an autonomous SMT-based constraint-solver for rational numbers [8] and the use of matrix interpretations *over the reals* [6]. Our benchmarks show that polynomials over the rationals are used in around 25% of the examples where a polynomial interpretation is required during the successful proof. Matrix interpretations are used in less than 4% of the proofs.

2.6 Termination Expert

In the (CS)DP framework, a strategy is applied to an initial (CSR, innermost CSR, ...) problem and returns a proof tree. This proof tree is later evaluated following a tree evaluation strategy (normally, breadth-first search).

With small differences depending on the particular kind of problem, we do the following:

1. We check the system for extra variables (at active positions) in the right-hand side of the rules.
2. We check whether the system is innermost equivalent (see Section 2.2). If it is true, then we transform the problem into an innermost one.
3. Then, we obtain the corresponding dependency pairs, obtaining a (CS)DP problem. And now, recursively:
   (a) Decision point between infinite processors and the *strongly connected component* (SCC) processor.
   (b) Subterm criterion processor.
   (c) Reduction triple (RT) processor with linear polynomials (LPoly) and coefficients in $N_2 = \{0, 1, 2\}$.
   (d) RT processor with LPoly and coefficients in $Q_2 = \{0, 1, 2, \frac{1}{2}\}$ and $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{1}{4}\}$ (in this order).
   (e) RT processor with simple mixed polynomials (SMPoly) and coefficients in N_2.
   (f) RT processor with SMPoly and rational coefficients in Q_2.
   (g) RT processor with 2-square matrices with entries in N_2 and Q_2.
   (h) Transformation processors (only twice to avoid nontermination of the strategy): instantiation, forward instantiation, and narrowing.
4. If the techniques above fail, then we use (CS)RPO.

The explanation of each processor can be found in [3,20]. Note also that all processors are new with respect to MU-TERM 4.3 [2].

2.7 External Use of MU-TERM

The Maude Termination Tool[4] (MTT [12]), which transforms proofs of termination of Maude programs into proofs of termination of CSR, use MU-TERM's expert as an external tool to obtain the proofs. The context-sensitive and order-sorted features developed as part of MU-TERM 5.0 are essential to successfully handling

[4] `http://www.lcc.uma.es/~duran/MTT`

Maude programs in MTT. The Knuth-Bendix completion tool $\mathsf{mkb}_{\mathrm{TT}}$ [29] is a modern completion tool that combines multi-completion with the use of termination tools. In the web version of the tool, the option to use MU-TERM as the external termination tool is available.

3 Conclusions

We have described MU-TERM 5.0, a new version of MU-TERM with new features for proving different termination properties like termination of *innermost* CSR, termination of *order-sorted rewriting* and termination of rewriting *modulo (associative or commutative) axioms*. Apart from that, a complete implementation of the *CSDP framework* [20] has been included in MU-TERM 5.0, leading to a much more powerful tool for proving termination of CSR. While transformations were used in MU-TERM 4.3, in MU-TERM 5.0 they are not used anymore. The research in the field has increased the number of examples which could be handled with CSDPs in 35 (see Table 1). Regarding proofs of *termination of rewriting*, from a collection of 1468 examples from the TPDB 7.0.2, MU-TERM 5.0 is able to prove (or disprove) termination of 835 of them. In contrast, MU-TERM 4.3 was able to deal with 503 only.

More details about these experimental results in all considered termination properties discussed in the previous sections can be found here:

`http://zenon.dsic.upv.es/muterm/benchmarks/index.html`

Thanks to the new developments reported in this paper, MU-TERM 5.0 has proven to be the most powerful tool for proving termination of CSR in the *context-sensitive* subcategory of the 2007, 2009, and 2010 editions of the International Competition of Termination Tools[5]. Moreover, in the *standard* subcategory, we have obtained quite good results in the 2009 and 2010 editions, being the third tool (among five) in solving more examples. We have also participated in the innermost category in the 2009 and 2010 editions and in the equational category in 2010.

Note also that MU-TERM 5.0 has a web interface that allows inexpert users to prove automatically termination by means of the 'automatic' option. This is very convenient for teaching purposes, for instance. And, apart from MTT, it is the only termination tool that accepts programs in OBJ/Maude syntax.

Therefore, MU-TERM 5.0 is no more a tool for proving termination of CSR only: We can say now that it has evolved to become a powerful termination tool which is able to prove termination of a wide range of interesting properties of rewriting with important applications to prove termination of programs in sophisticated rewriting-based programming languages like Maude or OBJ*.

[5] See `http://www.lri.fr/~marche/termination-competition/2007/`, where only AProVE and MU-TERM participated, and `http://termcomp.uibk.ac.at/termcomp/` where there were three more tools in the competition: AProVE, Jambox [13] (only in the 2009 edition), and VMTL [27]. AProVE and MU-TERM solved the same number of examples but MU-TERM was much faster. The 2008 edition had only one participant: AProVE.

References

1. Alarcón, B.: Innermost Termination of Context-Sensitive Rewriting. Master's thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain (2008)
2. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. Electronic Notes in Theoretical Computer Science 188, 105–115 (2007)
3. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. Information and Computation 208, 922–968 (2010)
4. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In: Konev, B., Wolter, F. (eds.) FroCos 2007. LNCS (LNAI), vol. 4720, pp. 73–87. Springer, Heidelberg (2007)
5. Alarcón, B., Lucas, S., Meseguer, J.: A Dependency Pair Framework for $A\vee C$-Termination. In: Ölveczky, P. (ed.) WRLA 2010. LNCS, vol. 6381, pp. 35–51. Springer, Heidelberg (2010)
6. Alarcón, B., Lucas, S., Navarro-Marset, R.: Proving Termination with Matrix Interpretations over the Reals. In: Geser, A., Waldmann, J. (eds.) Proc. of the 10th International Workshop on Termination, WST 2009, pp. 12–15 (2009)
7. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236(1-2), 133–178 (2000)
8. Borralleras, C., Lucas, S., Navarro-Marset, R., Rodríguez-Carbonell, E., Rubio, A.: Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. In: Schmidt, R.A. (ed.) CADE 2009. LNCS, vol. 5663, pp. 294–305. Springer, Heidelberg (2009)
9. Borralleras, C., Lucas, S., Rubio, A.: Recursive Path Orderings can be Context-Sensitive. In: Voronkov, A. (ed.) CADE 2002. LNCS (LNAI), vol. 2392, pp. 314–331. Springer, Heidelberg (2002)
10. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: All About Maude - A High-Performance Logical Framework. LNCS, vol. 4350. Springer, Heidelberg (2007)
11. Contejean, E., Marché, C., Monate, B., Urbain, X.: Proving Termination of Rewriting with CiME. In: Rubio, A. (ed.) Proc. of the 6th International Workshop on Termination, WST 2003, pp. 71–73 (2003)
12. Durán, F., Lucas, S., Meseguer, J.: MTT: The Maude Termination Tool (System Description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) IJCAR 2008. LNCS (LNAI), vol. 5195, pp. 313–319. Springer, Heidelberg (2008)
13. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewriting (2009), `http://joerg.endrullis.de/jambox.html`
14. Giesl, J., Middeldorp, A.: Innermost Termination of Context-Sensitive Rewriting. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS(LNAI), vol. 2450, pp. 231–244. Springer, Heidelberg (2003)
15. Giesl, J., Middeldorp, A.: Transformation Techniques for Context-Sensitive Rewrite Systems. Journal of Functional Programming 14(4), 379–427 (2004)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS (LNAI), vol. 4130, pp. 281–286. Springer, Heidelberg (2006)
17. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. Journal of Automatic Reasoning 37(3), 155–203 (2006)

18. Gnaedig, I.: Termination of Order-sorted Rewriting. In: Kirchner, H., Levi, G. (eds.) ALP 1992. LNCS, vol. 632, pp. 37–52. Springer, Heidelberg (1992)
19. Gramlich, B., Lucas, S.: Modular Termination of Context-Sensitive Rewriting. In: Proc. of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP 2002, pp. 50–61. ACM Press, New York (2002)
20. Gutiérrez, R., Lucas, S.: Proving Termination in the Context-Sensitive Dependency Pair Framework. In: Ölveczky, P.C. (ed.) WRLA 2010. LNCS, vol. 6381, pp. 18–34. Springer, Heidelberg (2010)
21. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. Journal of Functional and Logic Programming 1998(1), 1–61 (1998)
22. Lucas, S.: Termination of Rewriting With Strategy Annotations. In: Nieuwenhuis, R., Voronkov, A. (eds.) LPAR 2001. LNCS (LNAI), vol. 2250, pp. 666–680. Springer, Heidelberg (2001)
23. Lucas, S.: MU-TERM: A tool for proving termination of context-sensitive rewriting. In: van Oostrom, V. (ed.) RTA 2004. LNCS, vol. 3091, pp. 200–209. Springer, Heidelberg (2004)
24. Lucas, S.: Polynomials over the Reals in Proofs of Termination: from Theory to Practice. RAIRO Theoretical Informatics and Applications 39(3), 547–586 (2005)
25. Lucas, S.: Proving Termination of Context-Sensitive Rewriting by Transformation. Information and Computation 204(12), 1782–1846 (2006)
26. Lucas, S., Meseguer, J.: Order-Sorted Dependency Pairs. In: Antoy, S., Albert, E. (eds.) Proc. of the 10th International ACM SIGPLAN Sympsium on Principles and Practice of Declarative Programming, PPDP 2008, pp. 108–119. ACM Press, New York (2008)
27. Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 285–294. Springer, Heidelberg (2009)
28. Thiemann, R., Sternagel, C.: Loops under Strategies. In: Treinen, R. (ed.) RTA 2009. LNCS, vol. 5595, pp. 17–31. Springer, Heidelberg (2009)
29. Winkler, S., Sato, H., Middeldorp, A., Kurihara, M.: Optimizing $\mathsf{mkb}_{\mathrm{TT}}$. In: Lynch, C. (ed.) In: Lynch, C. (ed.) Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA 2010. Leibniz International Proceedings in Informatics (LIPIcs), vol. 6, pp. 373–384. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik (2010), http://drops.dagstuhl.de/opus/volltexte/2010/2664

18.10 Innermost Termination of Context-Sensitive Rewriting

10. B. Alarcón, and S. Lucas. **Innermost Termination of Context-Sensitive Rewriting**. Technical Report, http://hdl.handle.net/10251/10796, DSIC-ELP, UPV, 2011.

Innermost Termination of Context-Sensitive Rewriting

Beatriz Alarcón Salvador Lucas
DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, E-46022 Valencia, Spain
{balarcon , slucas}@dsic.upv.es

Abstract

Innermost context-sensitive rewriting (CSR) has been proved useful for modeling the computational behavior of programs of algebraic languages like Maude, OBJ, etc, which incorporate an innermost strategy which is used to break down the nondeterminism which is inherent to reduction relations. Furthermore, innermost termination of rewriting is often easier to prove than termination. Thus, under appropriate conditions, a useful strategy for proving termination of rewriting is trying to prove termination of innermost rewriting. This phenomenon has also been investigated for context-sensitive rewriting. Up to now, only few transformation-based methods have been proposed and used to (specifically) prove termination of innermost *CSR*. Powerful and efficient techniques for proving (innermost) termination of (unrestricted) rewriting like the dependency pair framework have not been considered yet. In this work, we investigate the adaptation of the dependency pair framework to innermost *CSR*. We provide a suitable notion of innermost context-sensitive dependency pair and show how to extend and adapt the main notions which conform the framework (chain, termination problem, processor, etc.). Thanks to the innermost context-sensitive dependency pairs, we can now use powerful techniques for proving termination of innermost *CSR*. This is made clear by means of some benchmarks showing that our techniques dramatically improve over previously existing transformational techniques, thus establishing the new state-of-the-art in the area. We have implemented them as part of the termination tool MU-TERM.

1 Introduction

Termination is one of the most interesting practical problems in computation and software engineering. A program or computational system is said to be *terminating* if it does not lead to any infinite computation for any possible call or input data. Ensuring termination is often a prerequisite for essential program properties like correctness. In the last years, many studies have been developed to analyze termination of programming languages, mainly of functional

[Gie95, LJB01, Xi02] and logic programming languages [CLS05, CT99, DD94, DL01, DS02, LMS03, Sma04]. In the case of imperative programming languages, it is becoming important in the last years [AAC+08, BMS05, CPR06, CS02, Tiw04]. Since most computational systems whose operational principle is based on reducing expressions can be described and analyzed by using notions and techniques coming from the abstract model of Term Rewriting Systems (TRSs [BN98, TeR03]), in many programming languages, it is possible to reduce the question of termination of programs to analyze termination of TRSs. For this reason, the development of techniques for proving termination of term rewrite systems becomes especially important since every improvement will have a positive impact on program verification of many programming languages. Following this approach, many powerful studies have been developed for both declarative and imperative programming languages. Regarding with termination of logic programs several works can be found: [AM93, KKS98, Mar94, Mar96, SGN09, SGST06]. Termination of the functional language Haskell [HPW92] has been developed quite recently [GSST06] and also termination of Java Bytecode [OBEG10]. Moreover, such computational systems (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) often incorporate a predefined reduction strategy which is used to break down the nondeterminism which is inherent to reduction relations. Eventually, this can rise problems, as each kind of strategy only behaves properly (i.e., it is normalizing, optimal, etc.) for particular classes of programs. One of the most commonly used strategy is the *innermost* one, in which only innermost redexes are reduced. Here, by an innermost redex we mean a redex containing no other redex. The innermost strategy corresponds to call by value or *eager* computation, that is, the computational mechanism of several programming languages where the arguments of a function are always evaluated before the application of the function which use them. It is well-known, however, that programs written in eager programming languages frequently run into a nonterminating behavior if the programs have not carefully been written to avoid such problems. For this reason, the designers of such eager programming languages have also developed some features and language constructs aimed at giving the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to arguments of symbols) have been used in programming languages such as Clean [NSEP92], Haskell [HPW92], Lisp [McC60], Maude [CDE+07], OBJ2 [FGJM85], OBJ3 [GWM+00], CafeOBJ [FN97], etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become 'more eager' and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become 'more lazy' thus (hopefully) avoiding nontermination.

Context-sensitive rewriting (*CSR* [Luc98, Luc02]) is a restriction of rewriting that forbids reductions on some subexpressions and that has proved useful to model and analyze such programming language features at different levels, see, e.g., [BM06, DLM+04, DLM+08, GM04, Luc01b, LM09]. Such a restriction of the rewriting computations is formalized at a very simple syntactic level:

that of the arguments of function symbols f in the signature $\mathcal{F}$. As usual, by a *signature* we mean a set of function symbols $f_1, \ldots, f_n, \ldots$ together with an *arity* function $ar : \mathcal{F} \to \mathbb{N}$ which establishes the number of 'arguments' associated to each symbol. A *replacement map* is a mapping $\mu : \mathcal{F} \to \wp(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each k-ary symbol f in the signature $\mathcal{F}$ [Luc98]. We use them to discriminate the argument positions on which the rewriting steps are allowed. In *CSR* we only rewrite μ-replacing subterms: every term t (as a whole) is μ-replacing by definition; and t_i (as well as all its μ-replacing subterms) is a μ-replacing subterm of $f(t_1, \ldots, t_k)$ if $i \in \mu(f)$.

Example 1 *Consider the following orthogonal TRS $\mathcal{R}$ which is a variant of an example in [Bor03]:*

$$\begin{array}{rcl}
\mathtt{from}(x) & \to & \mathtt{cons}(x, \mathtt{from}(\mathtt{s}(x))) \\
\mathtt{sel}(\mathtt{0}, \mathtt{cons}(x, xs)) & \to & x \\
\mathtt{sel}(\mathtt{s}(y), \mathtt{cons}(x, xs)) & \to & \mathtt{sel}(y, xs) \\
\mathtt{minus}(x, \mathtt{0}) & \to & x \\
\mathtt{minus}(\mathtt{s}(x), \mathtt{s}(y)) & \to & \mathtt{minus}(x, y) \\
\mathtt{quot}(\mathtt{0}, \mathtt{s}(y)) & \to & \mathtt{0} \\
\mathtt{quot}(\mathtt{s}(x), \mathtt{s}(y)) & \to & \mathtt{s}(\mathtt{quot}(\mathtt{minus}(x, y), \mathtt{s}(y))) \\
\mathtt{zWquot}(\mathtt{nil}, \mathtt{nil}) & \to & \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{cons}(x, xs), \mathtt{nil}) & \to & \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{nil}, \mathtt{cons}(x, xs)) & \to & \mathtt{nil} \\
\mathtt{zWquot}(\mathtt{cons}(x, xs), \mathtt{cons}(y, ys)) & \to & \mathtt{cons}(\mathtt{quot}(x, y), \mathtt{zWquot}(xs, ys))
\end{array}$$

together with $\mu(\mathtt{cons}) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for all other symbols f. According to [GM02a], innermost μ-termination of $\mathcal{R}$ implies its μ-termination as well. We will show how $\mathcal{R}$ can easily be proved innermost μ-terminating (and hence μ-terminating) by using the results in this paper.

The replacement map in Example 1 exemplifies one of the most typical applications of context-sensitive rewriting as a computational mechanism. The declaration $\mu(\mathtt{cons}) = \{1\}$ disallows reductions on the *list* part of the list constructor `cons`, thus making possible a kind of *lazy evaluation* of lists. We can still use projection operators as `sel` to continue the evaluation when needed. The other typical application is the declaration $\mu(\mathtt{if}) = \{1\}$ which allows us to forbid reductions on the two *alternatives* s and t of *if-then-else* expressions $\mathtt{if}(b, s, t)$ whereas it is still possible to perform reductions on the *boolean* part b, as required to implement the usual semantics of the operator.

Termination is also one of the most interesting problems when dealing with *CSR*. With *CSR* we can *achieve* a terminating behavior with nonterminating TRSs by pruning (all) infinite rewrite sequences.

Our focus is on termination of *innermost context-sensitive rewriting* (i.e., the variant of *CSR* where only the deepest μ-replacing redexes are contracted). Termination of innermost context-sensitive rewriting has been proved useful for proving termination of programs in programming languages like Maude and

OBJ* which permit to control the program execution by means of such context-sensitive annotations [Luc01a, Luc01b]. Techniques for proving termination of innermost *CSR* were first investigated in [GM02b, Luc01a]. These papers, though, only consider *transformational* techniques, where the original CS-TRS $(\mathcal{R}, \mu)$ is transformed into a TRS $\mathcal{R}^{\mu}_{\Theta}$ (where Θ represents the transformation which has been used) whose *innermost* termination implies the innermost termination of *CSR* for $(\mathcal{R}, \mu)$. The *dependency pairs method* [AG00, GAO02, GTS04, GTSF06, HM04, HM05], one of the most powerful techniques for proving termination of rewriting, had not been investigated in connection with proofs of termination of *CSR* until [AGL06]. As shown in [AGL07], proofs of termination using context-sensitive dependency pairs (CSDPs) are much more powerful and faster than any other technique for proving termination of *CSR*. As we show here, dealing with innermost *CSR*, we have a similar situation.

Proving innermost termination of rewriting is often easier than proving termination of rewriting [AG00] and, for some relevant classes of TRSs, innermost termination of rewriting is even equivalent to termination of rewriting [Gra95, Gra96]. In [GM02b, GL02a] it is proved that the equivalence between termination of innermost *CSR* and termination of *CSR* holds in some interesting cases (e.g., for *orthogonal* CS-TRSs).

During the last years, we have investigated in deep how to prove termination of context-sensitive rewriting by using dependency pairs, since they have proven to be one of the most powerful techniques for proving termination of unrestricted rewriting. In [AGL10], we define the notion of context-sensitive dependency pairs following the approach of [HM04] which consists of considering the structure of the infinite rewrite sequences starting from minimal non-terminating terms. Therefore, all the advantages and improvements over this research can also be taken into account in innermost context-sensitive rewriting, improving our previous results on this field in [AL07].

1.1 Plan of the paper

After some preliminaries in Section 2, we develop the material in the paper in three main parts:

1. We investigate the structure of infinite innermost context-sensitive rewrite sequences. This analysis is essential to provide an appropriate definition of innermost context-sensitive dependency pair, and the related notions of innermost chains, graph, etc. Section 3 provides appropriate notions of *minimal* innermost non-μ-terminating terms and introduces the main properties of such terms. It also recalls the notion of *hidden term* in a CS-TRS. This notion turns to be essential for the appropriate treatment of our dependency pairs. We investigate the structure of infinite innermost context-sensitive rewrite sequences starting from strongly minimal innermost non-μ-terminating terms.

2. We define the notions of *innermost context-sensitive dependency pair* and *innermost context-sensitive chain of pairs* and show how to use them to

characterize innermost termination of *CSR*. Sections 4 introduces the general framework to compute and use innermost context-sensitive dependency pairs for proving innermost termination of *CSR*. The introduction of a new kind of dependency pairs (the *collapsing* dependency pairs) leads to a notion of innermost context-sensitive dependency *chain*, which is quite different from the standard one. We prove that our *innermost context-sensitive dependency pair approach* fully characterizes termination of innermost *CSR*.

3. We describe a suitable *framework* for dealing with proofs of termination of innermost *CSR* by using the previous results. Section 5 provides an adaptation of the *dependency pair framework* [GTS04, GTSF06] to innermost *CSR* by defining appropriate notions of *CS problem* and *CS processor* which rely in the notions and results investigated in the second part of the paper. Section 6 introduces several basic processors for proving innermost termination of *CSR*. Section 7 introduces the notion of *innermost context-sensitive (dependency) graph* and the associated CS processor which formalizes the usual practice of analyzing the absence of infinite (minimal) innermost chains by considering the (maximal) cycles in the dependency graph. As in the standard case, the ICS-dependency graph is not computable, so we show how to obtain the *estimated* ICS-dependency graph which is a computable overestimation of it. Section 8 adapts the notion of usable rules to deal with proofs of innermost *CSR* by using term orderings. We introduce the notion of *μ-reduction pair*, which is the straightforward adaptation of reduction pairs used for dealing with dependency pairs in the standard case. Section 9 adapts to the context-sensitive setting, the notion of *usable argument* introduced by Férnandez [Fer05] to prove innermost termination of rewriting by proving termination of *CSR*. In this way, we can prove innermost termination of *CSR* by proving innermost termination of *CSR* using a *more restrictive* replacement map. We also include this criterion as a processor in the innermost context-sensitive dependency pair framework. Section 10 adapts *narrowing transformation* of pairs in [GTSF06] to innermost *CSR* and the new framework.

The paper ends with an experimental evaluation of our techniques in Section 11. Section 12 concludes.

2 Preliminaries

This section collects a number of definitions and notations about term rewriting. More details and missing notions can be found in [BN98, Ohl02, TeR03].

Let A be a set and $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$ be a binary relation on A. We denote the transitive closure of R by R^+ and its reflexive and transitive closure by R^*. We say that R is *terminating* (*strongly normalizing*) if there is no infinite sequence $a_1 \mathsf{R} a_2 \mathsf{R} a_3 \cdots$. A reflexive and transitive relation R is a quasi-ordering.

2.1 Signatures, Terms, and Positions

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and $\mathcal{F}$ denotes a signature, i.e., a set of function symbols $\{\mathtt{f}, \mathtt{g}, \ldots\}$, each having a fixed arity given by a mapping $ar : \mathcal{F} \to \mathbb{N}$. The set of terms built from $\mathcal{F}$ and $\mathcal{X}$ is $\mathcal{T}(\mathcal{F}, \mathcal{X})$. A term is *ground* if it contains no variable. A term is said to be *linear* if it has no multiple occurrences of a single variable.

Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. We denote the empty chain by Λ. Given positions p, q, we denote their concatenation as $p.q$. Positions are ordered by the standard prefix ordering: $p \leq q$ if $\exists q'$ such that $q = p.q'$ If p is a position, and Q is a set of positions, $p.Q = \{p.q \mid q \in Q\}$. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of nonvariable symbols in t are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$, and $\mathcal{P}os_{\mathcal{X}}(t)$ are the positions of variables. The subterm at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s.

We write $t \trianglerighteq s$, read *s is a subterm of t*, if $s = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright s$ if $t \trianglerighteq s$ and $t \neq s$. We write $t \not\trianglerighteq s$ and $t \not\triangleright s$ for the negation of the corresponding properties. The symbol labeling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with a 'hole' $\Box$ (a fresh constant symbol). We write $C[\,]_p$ to denote that there is a (usually single) hole $\Box$ at position p of C. Generally, we write $C[\,]$ to denote an arbitrary context and make explicit the position of the hole only if necessary. $C[\,] = \Box$ is called the *empty* context.

2.2 Substitutions

A substitution is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$. Denote as ε the 'identity' substitution: $\varepsilon(x) = x$ for all $x \in \mathcal{X}$. The set $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ is called the *domain* of σ.

Remark 1 *In this paper, we do* not *impose that the domain of the substitutions is finite. This is usual practice in the dependency pair approach, where a single substitution is used to instantiate an infinite number of variables coming from renamed versions of the dependency pairs (see below).*

Whenever $\mathcal{D}om(\sigma) \cap \mathcal{D}om(\sigma') = \varnothing$, for substitutions σ, σ', we denote by $\sigma \cup \sigma'$, a substitution such that $(\sigma \cup \sigma')(x) = \sigma(x)$ if $x \in \mathcal{D}om(\sigma)$ and $(\sigma \cup \sigma')(x) = \sigma'(x)$ if $x \in \mathcal{D}om(\sigma')$.

2.3 Renamings and unifiers

A *renaming* is an injective substitution ρ such that $\rho(x) \in \mathcal{X}$ for all $x \in \mathcal{X}$. For renamings, we assume that $\mathcal{V}ar(\rho)$ is finite (which is the usual practice) and also idempotency, i.e., $\rho(\rho(x)) = \rho(x)$ for all $x \in \mathcal{X}$.

The quasi-ordering of subsumption $\leq$ over $\mathcal{T}(\mathcal{F}, \mathcal{X})$ is $t \leq t' \Leftrightarrow \exists \sigma.\ t' = \sigma(t)$. We denote as $\sigma \leq \sigma'$ the fact that $\sigma(x) \leq \sigma'(x)$ for all $x \in \mathcal{X}$, thus extending the quasi-ordering to substitutions.

A substitution σ such that $\sigma(s) = \sigma(t)$ for two terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is called a *unifier* of s and t; we also say that s and t unify (with substitution σ). If two terms s and t unify, then there is a unique (up to renaming of variables) *most general unifier (mgu)* θ which is *minimal* (w.r.t. the subsumption quasi-ordering $\leq$) among all other unifiers of s and t.

A relation $\mathsf{R} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X}) \times \mathcal{T}(\mathcal{F}, \mathcal{X})$ on terms is *stable* if for all terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, and substitutions σ, we have $\sigma(s)\,\mathsf{R}\,\sigma(t)$ whenever $s\,\mathsf{R}\,t$.

2.4 Rewrite Systems and Term Rewriting

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l and r is the right-hand side (*rhs*). A rewrite rule $l \to r$ is said to be *collapsing* if $r \in \mathcal{X}$. A *Term Rewriting System* (TRS) is a pair $\mathcal{R} = (\mathcal{F}, R)$, where R is a set of rewrite rules. Given TRSs $\mathcal{R} = (\mathcal{F}, R)$ and $\mathcal{R}' = (\mathcal{F}', R')$, we let $\mathcal{R} \cup \mathcal{R}'$ be the TRS $(\mathcal{F} \cup \mathcal{F}', R \cup R')$. An instance $\sigma(l)$ of a *lhs* l of a rule is called a *redex*. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$, called *constructors* and symbols $f \in \mathcal{D}$, called *defined functions*, where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

Example 2 *Consider again the TRS in Example 1. The symbols* `from`*,* `sel`*,* `minus`*,* `quot` *and* `zWquot` *are defined, and* `s`*,* `0`*,* `cons`*, and* `nil` *are constructors.*

For simplicity, we often write $l \to r \in \mathcal{R}$ instead of $l \to r \in R$ to express that the rule $l \to r$ is a rule of $\mathcal{R}$. The pair $\langle \sigma(l)[\sigma(r')]_p, \sigma(r) \rangle$ is called a critical pair and is also called an overlay if $p = \Lambda$. A critical pair $\langle t, s \rangle$ is trivial if $t = s$. The critical pairs of a TRS $\mathcal{R}$ are the critical pairs between any two of its (renamed) rules; this includes overlaps of a rule with a renamed variant of itself, except at the root, i.e., if $p = \Lambda$. A TRS $\mathcal{R}$ is left-linear if for all $l \to r \in R$, l is a linear term. A left-linear TRS without critical pairs is called orthogonal. A term $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ rewrites to s (at position p), written $t \xrightarrow{p}_{\mathcal{R}} s$ (or just $t \to s$, or $t \to_{\mathcal{R}} s$), if $t|_p = \sigma(l)$ and $s = t[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(t)$ and substitution σ. We write $t \xrightarrow{>p}_{\mathcal{R}} s$ if $t \xrightarrow{q}_{\mathcal{R}} s$ for some $q > p$. A TRS $\mathcal{R}$ is terminating if its one step rewrite relation $\to_{\mathcal{R}}$ is terminating.

2.5 Innermost rewriting

A term is a normal form if it contains no redex. A substitution σ is normalized if $\sigma(x)$ is a normal form for all $x \in \mathcal{D}om(\sigma)$. A term $f(t_1, \ldots, t_k)$ is argument normalized if t_i is a normal form for all $1 \leq i \leq n$. An innermost redex is an argument normalized redex. A term s rewrites innermost to t, written $s \to_i t$, if $s \to t$ at position p and $s|_p$ is an innermost redex. Let $\mathcal{R}$ be a TRS. For any symbol f let $Rules(\mathcal{R}, f)$ be the set of rules $l \to r$ defining f and such that the left-hand sides l are argument normalized. For any term t the set of usable rules $\mathbf{U}(\mathcal{R}, t)$ is as follows:

$$\begin{array}{lcl} \mathbf{U}(\mathcal{R}, x) & = & \varnothing \\ \mathbf{U}(\mathcal{R}, f(t_1, \ldots, t_n)) & = & \mathit{Rules}(\mathcal{R}, f) \cup \bigcup_{i \in ar(f)} \mathbf{U}(\mathcal{R}', t_i) \cup \bigcup_{l \to r \in \mathit{Rules}(\mathcal{R}, f)} \mathbf{U}(\mathcal{R}', r) \end{array}$$

where $\mathcal{R}' = \mathcal{R} - \mathit{Rules}(\mathcal{R}, f)$.

2.6 (Innermost) Context-Sensitive Rewriting

A mapping $\mu : \mathcal{F} \to \wp(\mathbb{N})$ is a *replacement map* (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F},\ \mu(f) \subseteq \{1, \ldots, ar(f)\}$ [Luc98]. Let $M_{\mathcal{F}}$ be the set of all $\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $(\mathcal{F}, R)$). Let $\mu_\top$ be the replacement map given by $\mu_\top(f) = \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}$ (i.e., no replacement restrictions are specified).

A binary relation R on terms is μ-monotonic if whenever $t\ \mathsf{R}\ s$ we have that $f(t_1, \ldots, t_{i-1}, t, \ldots, t_k)\ \mathsf{R}\ f(t_1, \ldots, t_{i-1}, s, \ldots, t_k)$ for all $f \in \mathcal{F}$, $i \in \mu(f)$, and $t, s, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$. If R is $\mu_\top$-monotonic, we just say that R is *monotonic*.

The set of *μ-replacing positions* $\mathcal{P}os^\mu(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^\mu(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^\mu(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^\mu(t|_i)$, if $t \notin \mathcal{X}$. When no replacement map is made explicit, the μ-replacing positions are often called *active*; and the non-μ-replacing ones are often called *frozen*. The following result about *CSR* is often used without any explicit mention.

Proposition 1 [Luc98] *Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and $p = q.q' \in \mathcal{P}os(t)$. Then $p \in \mathcal{P}os^\mu(t)$ iff $q \in \mathcal{P}os^\mu(t) \wedge q' \in \mathcal{P}os^\mu(t|_q)$*

The μ-replacing subterm relation $\trianglerighteq_\mu$ is given by $t \trianglerighteq_\mu s$ if there is $p \in \mathcal{P}os^\mu(t)$ such that $s = t|_p$. We write $t \triangleright_\mu s$ if $t \trianglerighteq_\mu s$ and $t \neq s$. We write $t \triangleright_{\not\mu} s$ to denote that s is a non-μ-replacing (hence strict) subterm of t: $t \triangleright_{\not\mu} s$ if there is $p \in \mathcal{P}os(t) - \mathcal{P}os^\mu(t)$ such that $s = t|_p$. The set of *μ-replacing* variables of a term t, i.e., variables occurring at some μ-replacing position in t, is $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid t \trianglerighteq_\mu x\}$. The set of non-$\mu$-replacing variables of t, i.e., variables occurring at some non-μ-replacing position in t, is $\mathcal{V}ar^{\not\mu}(t) = \{x \in \mathcal{V}ar(t) \mid t \triangleright_{\not\mu} x\}$. Note that $\mathcal{V}ar^\mu(t)$ and $\mathcal{V}ar^{\not\mu}(t)$ do not need to be disjoint.

A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a CS-TRS. In *context-sensitive rewriting*, we (only) contract *μ-replacing* redexes: t μ-rewrites to s, written $t \hookrightarrow_\mu s$ (or $t \hookrightarrow_{\mathcal{R},\mu} s$ and even $t \hookrightarrow s$), if $t \xrightarrow{p}_{\mathcal{R}} s$ and $p \in \mathcal{P}os^\mu(t)$.

Example 3 *Consider $\mathcal{R}$ and μ as in Example 1. Then, we have:*

$$\underline{\mathtt{from(0)}} \hookrightarrow_\mu \mathtt{cons(0, \underline{from(s(0))})} \not\hookrightarrow_\mu \mathtt{cons(0, cons(s(0), from(s(s(0))))}$$

Since the second argument of `cons` *is not μ-replacing, we have that $2 \notin \mathcal{P}os^\mu(\mathtt{cons(0, from(s(0)))})$, and the redex* `from(s(0))` *cannot be μ-rewritten.*

A term t is μ-terminating (or $(\mathcal{R}, \mu)$-terminating, if we want an explicit reference to the involved TRS $\mathcal{R}$) if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_\mu t_2 \hookrightarrow_\mu \cdots \hookrightarrow_\mu t_n \hookrightarrow_\mu \cdots$ starting from t. A TRS $\mathcal{R}$ is μ-terminating if $\hookrightarrow_\mu$ is terminating.

A μ-normal form is a term which cannot be μ-rewritten. Let $\mathsf{NF}_\mu(\mathcal{R})$ (or just NF_μ if no confusion arises) be the set of μ-normal forms of a TRS $\mathcal{R}$.

A substitution σ is μ-normalized if $\sigma(x)$ is a μ-normal form for all $x \in \mathcal{D}om(\sigma)$. A term $t = f(t_1, \ldots, t_k)$ is argument μ-normalized if t_i is a μ-normal form for all $i \in \mu(f)$. A μ-innermost redex is an argument μ-normalized redex, i.e., $t = \sigma(l)$ for some substitution σ and rule $l \to r \in \mathcal{R}$ and for all $p \in \mathcal{P}os^\mu(t - \Lambda)$, $t|_p \in \mathsf{NF}_\mu$. A term s innermost μ-rewrites to t, written $s \hookrightarrow_{\mathsf{i}} t$, if $s \xrightarrow{p}_{\mathcal{R}} t$, $p \in \mathcal{P}os^\mu(s)$, and $s|_p$ is a μ-innermost redex. Let innermost μ-rewriting below the root be $\overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} = (\overset{>\Lambda}{\hookrightarrow} \cap \hookrightarrow_{\mathsf{i}})$. Termination of *CSR* is fully captured by the so-called μ-reduction orderings, i.e., well-founded, stable orderings $\sqsupset$ which are μ-monotonic. A TRS $\mathcal{R}$ is innermost μ-terminating if $\hookrightarrow_{\mu,\mathsf{i}}$ is terminating. We write $s \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} t$ if $s \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} t$ and $t \in \mathsf{NF}_\mu$.

A term t *μ-narrows* to a term s (written $t \rightsquigarrow_{\mathcal{R},\mu,\theta} s$), if there is a nonvariable μ-replacing position $p \in \mathcal{P}os^\mu_{\mathcal{F}}(t)$ and a rule $l \to r$ in $\mathcal{R}$ (sharing no variable with t) such that $t|_p$ and l unify with most general unifier θ and $s = \theta(t[r]_p)$.

3 Minimal innermost non-μ-terminating terms and Infinite Innermost μ-rewrite Sequences

In the following, we show how to adapt our results about the structure of infinite context-sensitive rewrite sequences [AGL10, Section 3] to the innermost sequences. Most proofs are only slightly different from the original ones and therefore we comment on the differences only (for full proofs see [Ala08]). Major differences come from particularities of reductions under an innermost strategy. In some cases, they bring us some advantages over the case of 'free' reductions in *CSR*. In the following we discuss some of these peculiarities. In the innermost (context-sentive) setting, matching substitutions are always (μ-)normalized. According to the discussion in [AGL10], we introduce the following:

Definition 1 ((Strongly) minimal innermost non-μ-terminating term) *Let $\mathcal{M}_{\infty,\mu,\mathsf{i}}$ be the set of minimal innermost non-μ-terminating terms in the following sense: t belongs to $\mathcal{M}_{\infty,\mu,\mathsf{i}}$ if t is not innermost μ-terminating and every strict μ-*replacing *subterm s of t (i.e., $t \rhd_\mu s$) is innermost μ-terminating. Let $\mathcal{T}_{\infty,\mu,\mathsf{i}}$ be a set of strongly minimal innermost non-μ-terminating terms in the following sense: t belongs to $\mathcal{T}_{\infty,\mu,\mathsf{i}}$ if t is innermost non-μ-terminating and every strict subterm u (i.e., $t \rhd u$) is innermost μ-terminating. It is obvious that $root(t) \in \mathcal{D}$ for all $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$ or $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*

Note that $\mathcal{T}_{\infty,\mu,\mathsf{i}} \subseteq \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Before starting our discussion about minimal innermost non-μ-terminating terms, we provide auxiliary results about innermost μ-terminating terms (see [AGL10, Lemmata 1,2,3,4]).

Proposition 2 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS, $\mu \in M_{\mathcal{F}}$, and $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$.*

1. *If s is innermost μ-terminating and $s \trianglerighteq_\mu t$ or $s \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} t$ then t is innermost μ-terminating.*

2. *If s is not innermost μ-terminating, then there is a subterm t of s ($s \trianglerighteq t$) such that $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$. Furthermore, there is a μ-replacing subterm t of s ($s \trianglerighteq_\mu t$) such that $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*

3. *If $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, $t \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} u$ and u is not innermost μ-terminating, then $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$.*

The following result is the innermost context-sensitive version of Lemma 1 in [HM04] that uses previous results. This proposition establishes that, given a minimal not innermost μ-terminating term $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there are only two ways for an infinite innermost μ-rewrite sequence to proceed. The first one is by using 'visible' parts of the rules which correspond to μ-replacing nonvariable subterms in the right-hand sides which are rooted by a defined symbol. This would corresponds with the straightforward extension of the original result but taking into account the reemplacement restrictions. The second one is by showing up 'hidden' not innermost μ-terminating subterms which are activated by *migrating* variables in a rule $l \to r$, i.e., variables $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)$ which are *not* μ-replacing in the left-hand side l but become μ-replacing in the right-hand side r.

Proposition 3 *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Then for all $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there exist $l \to r \in R$, a substitution σ such that $\sigma(l)$ is argument μ-normalized and a term $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ such that $t \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} \sigma(l) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma(r) \trianglerighteq_\mu u$ and either*

1. *there is a nonvariable μ-replacing subterm s of r, $r \trianglerighteq_\mu s$, such that $u = \sigma(s)$ and $\sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$ for all $x \in \mathcal{V}ar(s) \cap \mathcal{V}ar^\mu(l)$, or*

2. *there is $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)$ such that $\sigma(x) \trianglerighteq_\mu u$, that is, $\sigma(x) = C[u]_p$ for some context $C[]_p$ with $p \in Pos^\mu(C[]_p)$.*

Proof.

Consider an infinite innermost μ-rewrite sequence starting from t. By definition of $\mathcal{M}_{\infty,\mu,\mathsf{i}}$, all proper μ-replacing subterms of t are innermost μ-terminating. Therefore, t has an inner reduction (of innermost μ-rewriting steps) to an instance $\sigma(l)$ of the left-hand side of a rule $l \to r$ of $\mathcal{R}$, such that no strict μ-replacing subterm of $\sigma(l)$ is a redex, i.e. $\sigma(l)$ is argument μ-normalized. Then we have $t \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} \sigma(l) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma(r)$ and $\sigma(r)$ is not innermost μ-terminating. Note that, $\sigma(l)$ must be argument μ-normalized; otherwise, the last step would not be an innermost μ-rewriting step. Thus, we can write $t = f(t_1, \ldots, t_k)$ and $\sigma(l) = f(l_1, \ldots, l_k)$ for some k-ary defined symbol f, and $t_i \hookrightarrow^*_{\mathsf{i}} \sigma(l_i)$ for all i, $1 \leq i \leq k$. More precisely, $t_i \overset{!}{\hookrightarrow}_{\mathsf{i}} \sigma(l_i)$ if $i \in \mu(f)$ Since $\sigma(l)$ is argument μ-normalized, $\sigma(x) \in \mathsf{NF}_\mu$ for all μ-replacing variables x in l: $x \in \mathcal{V}ar^\mu(l)$. Since $\sigma(r)$ is not innermost μ-terminating, by Proposition 2-2 it contains a μ-replacing subterm $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$: $\sigma(r) \trianglerighteq_\mu u$, i.e., there is a position $p \in \mathcal{P}os^\mu(\sigma(r))$ such that $\sigma(r)|_p = u$. We consider two cases:

1. If $p \in \mathcal{P}os_{\mathcal{F}}(r)$ is a nonvariable position of r, then there is a μ-replacing subterm s of r, such that $u = \sigma(s)$. Note that $\sigma(x) \in \mathsf{NF}_\mu$ for all $x \in \mathcal{V}ar(r) \setminus \mathcal{V}ar^\mu(l)$.

2. If $p \notin \mathcal{P}os_{\mathcal{F}}(r)$, then there is a μ-replacing variable position $q \in \mathcal{P}os^\mu(r) \cap \mathcal{P}os_{\mathcal{X}}(r)$ such that $q \leq p$. Let $x \in \mathcal{V}ar^\mu(r)$ be such that $r|_q = x$. Then, $\sigma(x) \trianglerighteq_\mu u$ and $\sigma(x)$ is not innermost μ-terminating (by assumption, $u \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ is not innermost μ-terminating: by Proposition 2-1, $\sigma(x)$ cannot be innermost μ-terminating either). Since $\sigma(l_i)$ is innermost μ-terminating for all $i \in \mu(f)$, and $\sigma(x) \in \mathsf{NF}_\mu$ for all μ-replacing variables in l , we conclude that $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)$.

□

Proposition 3 entails the following result, which establishes some properties of infinite sequences starting from minimal innermost non-μ-terminating terms.

Corollary 1 *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. For all $t \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, there is an infinite sequence*

$$t \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} \cdots$$

where, for all $i \geq 1$, $l_i \to r_i \in R$ are rewrite rules, σ_i are substitutions, $\sigma_i(l_i)$ is argument μ-normalized, and terms $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ are minimal innermost non-μ-terminating terms such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable subterm s_i such that $r_i \trianglerighteq_\mu s_i$ and $\sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$ for all $x \in \mathcal{V}ar(s_i) \cap \mathcal{V}ar^\mu(l_i)$, or*

2. *$\sigma_i(x_i) \trianglerighteq_\mu t_i$ which is equivalent to $\sigma(x) = C[t_i]_{p_i}$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$ and context $C[]_{p_i}$ with $p_i \in Pos^\mu(C[]_{p_i})$.*

Now we pay attention to Item 2 of Proposition 3. To analyze in deep infinite sequences starting from minimal innermost non-μ-terminating terms we need to go inside the instantiation of the migrating variable, $\sigma(x)$. Since in (innermost) context-sensitive rewriting, function calls can be delayed, terms that are (innermost) μ-terminating can generate future (innermost) non-μ-terminating subterms. By Lemma 2 we know that innermost μ-termination is preserved under μ-rewritings and extraction of μ-replacing subterms, therefore, these innermost non-μ-terminating subterms introduced by innermost μ-rewriting steps can only occur at frozen positions in the reducts. This is captured by the notion of *hidden term*.

Definition 2 (Hidden Term [AGL10]) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. We say that $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ is a* hidden term *if there is a rule $l \to r \in R$ such that $r \rhd_\mu t$. Let $\mathcal{HT}(\mathcal{R}, \mu)$ (or just $\mathcal{HT}$, if $\mathcal{R}$ and μ are clear for the context) be the set of all hidden terms in $(\mathcal{R}, \mu)$. We say that $f \in \mathcal{F}$ is a* hidden symbol *if it occurs in a hidden term. Let $\mathcal{H}(\mathcal{R}, \mu)$ (or just $\mathcal{H}$) be the set of all hidden symbols in $(\mathcal{R}, \mu)$. We also use $\mathcal{DHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT} \mid root(t) \in \mathcal{D}\}$ for the set of hidden terms which are rooted by a* defined *symbol.*

Example 4 *For $\mathcal{R}$ and μ as in Example 1, the hidden terms are* $\mathtt{from}(\mathtt{s}(x))$, $\mathtt{s}(x)$, *and* $\mathtt{zWquot}(zs, ys)$. *The hidden symbols are* `from`, `s` *and* `zWquot`. *Finally,* $\mathcal{DHT}(\mathcal{R}, \mu) = \{\mathtt{from}(\mathtt{s}(x)), \mathtt{zWquot}(zs, ys)\}$.

Innermost non-μ-terminating terms at frozen positions can be activated by some specific contexts. In Proposition 3 (2), the intended role of hidden terms in the binding of the migrating variable $\sigma(x) = C[u]_p$ is that u' is a hidden term such that $\theta(u') = u$ for some substitution θ and context $C[]_p$. This context can only be composed by symbols f contained in hidden terms $f(\ldots, r_i, \ldots)$ such that $r' \rhd_{\not\mu} f(\ldots, r_i, \ldots) \trianglerighteq_\mu r_i$ for a rule $l' \to r' \in \mathcal{R}$ satisfying:

- r_i is a nonvariable term and $\sigma(r_i) = u$, or
- r_i is a variable at a frozen position in both, l and r.

These symbols conforms what is called as hiding context.

First notion of hiding context was found in [AEF+08] but it has been recently slightly redefined in [GL10]. We follow the last definition since it present some advantages.

Definition 3 (Hiding Context [GL10]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. A function symbol f* hides position $i \in \mu(f)$ in the rule $l \to r \in \mathcal{R}$ *if $r \rhd_{\not\mu} f(r_1, \ldots, r_n)$ for some terms $r_1, \ldots, r_n$, and r_i contains a μ-replacing defined symbol (i.e., $\mathcal{P}os^\mu_\mathcal{D}(r_i) \neq \varnothing$) or a variable $x \in (\mathcal{V}ar^{\not\mu}(l) \cap \mathcal{V}ar^{\not\mu}(r)) \setminus (\mathcal{V}ar^\mu(l) \cup \mathcal{V}ar^\mu(r))$ which is μ-replacing in r_i (i.e., $x \in \mathcal{V}ar^\mu(r_i)$). We say that f hides position i in $\mathcal{R}$ if there is a rule $l \to r$ such that f hides position i in $l \to r$. A context $C[\Box]$ is* hiding *if*

1. *$C[\Box] = \Box$, or*
2. *$C[\Box] = f(t_1, \ldots, t_{i-1}, C'[\Box], t_{i+1}, \ldots, t_k)$, where f hides position i and $C'[\Box]$ is a hiding context.*

These notions are used and combined to model infinite context-sensitive rewrite sequences starting from strongly minimal innermost non-μ-terminating, although, first, we need some previous results.

Definition 4 (Hiding Property [AEF+08]) *A term u has the* hiding property *iff*

- *$u \in \mathcal{M}_{\infty,\mu,\mathrm{i}}$ and*
- *whenever $u \rhd_{\not\mu} s \trianglerighteq_\mu t'$ for some terms s and t' with $t' \in \mathcal{M}_{\infty,\mu,\mathrm{i}}$, then t' is an instance of a hidden term and $s = C[t']$ for some hiding context $C[\,]$.*

Lemma 1 ([AEF+08]) *Let u be a term with the hiding property and let $u \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} v \trianglerighteq_\mu w$ with $w \in \mathcal{M}_{\infty,\mu,\mathrm{i}}$. Then w also has the hiding property.*

The proof of the previous lemma differs from the one in [AEF+08] in the refinement done in the notion of hiding context mentioned in [GL10] and it is slightly different from the one in [GL10] since we are dealing with innermost rewriting and all μ-replacing variables of the instantiated left-hand sides of the rules applied in a innermost μ-rewrite sequence are in μ-normal form: no matter if they are in a nonactive position on the right-hand side, they cannot start any reduction. In [GL10] it is not necessary either since in a μ-rewrite sequence, these variables could start a reduction but due to minimality, these reductions would be finite.

In the following, we consider a function $\textsc{Ren}^\mu$ [AGL06, AGL10] which *independently* renames all *occurrences* of μ-replacing variables within a term t by using new fresh variables which are not in $\mathcal{V}ar(t)$. Note that $\textsc{Ren}^\mu(t)$ keeps variables at non-μ-replacing positions untouched.

Proposition 4 ([AGL10]) *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution. If $\sigma(t) \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma(l)$ for some (probably renamed) rule $l \to r \in R$, then $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

Corollary 2 ([AGL10]) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$ be a nonvariable term and σ be a substitution such that $\sigma(t) \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Then, $\textsc{Ren}^\mu(t)$ is μ-narrowable.*

In the following, we write $\textsc{Narr}^\mu(t)$ [AGL10] to indicate that t is μ-narrowable (w.r.t. the intended TRS $\mathcal{R}$). We also let

$$\mathcal{NHT}(\mathcal{R}, \mu) = \{t \in \mathcal{DHT} \mid \textsc{Narr}^\mu(\textsc{Ren}^\mu(t))\}$$

be the set of *hidden terms* which are rooted by a *defined* symbol, and that, after applying $\textsc{Ren}^\mu$, become *μ-narrowable*.

As a consequence of the previous results, we have the following main result.

Theorem 1 (Minimal Innermost Sequence) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. For all $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$, there is an infinite sequence*

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}{}^*_{\mathsf{i}} \cdots$$

where, for all $i \geq 1$, $l_i \to r_i \in R$, σ_i is a substitution, $\sigma_i(l_i)$ is argument μ-normalized, and $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ is a minimal innermost non-μ-terminating term such that either

1. *$t_i = \sigma_i(s_i)$ for some nonvariable term s_i such that $r_i \trianglerighteq_\mu s_i$, or*
2. *$\sigma_i(x_i) = \theta_i(C_i[t'_i])$ and $t_i = \theta_i(t'_i)$ for some variable $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$, $t'_i \in \mathcal{NHT}(\mathcal{R}, \mu)$, hiding context $C_i[\Box]$, and substitution θ_i.*

PROOF.

Since $\mathcal{T}_{\infty,\mu,\mathsf{i}} \subseteq \mathcal{M}_{\infty,\mu,\mathsf{i}}$, by Corollary 1, we have a sequence

$$t = t_0 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^{*} \cdots$$

where, for all $i \geq 1$, $l_i \to r_i \in R$, σ_i is a substitution such that $\sigma(l_i)$ is argument μ-normalized, $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, and either (1) $t_i = \sigma_i(s_i)$ for some s_i such that $r_i \trianglerighteq_\mu s_i$ or (2) $\sigma_i(x_i) \trianglerighteq_\mu t_i$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$ (and hence $\sigma(l_i) \rhd_\mu t_i$ and $\sigma(r_i) \trianglerighteq_\mu t_i$ as well). If $\sigma_i(x_i) \trianglerighteq_\mu t_i$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(l_i)$, it means that $\sigma(l_i) \rhd_\mu C_i[t_i]$. Since $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$, it has the hiding property and, by Lemma 1, all $\sigma(l_i)$ satisfies the *hiding* property. Hence, $C_i[t_i] = \theta_i(C'_i[t'_i])$ where $t'_i \in \mathcal{DHT}(\mathcal{R},\mu)$ and $C'_i[\,]$ is a hiding context. By Corollary 2 we have $t'_i \in \mathcal{NHT}$. □

4 Innermost Context-Sensitive Dependency Pairs and Chains

An essential property of the dependency pairs method is that it provides a *characterization* of termination of TRSs $\mathcal{R}$ as the absence of infinite (minimal) *chains of dependency pairs* [AG00, GTSF06]. As we prove here this is also true for innermost *CSR*. First, we have to introduce a suitable notion of innermost dependency pair and chain which can be used for this purpose.

In innermost *CSR*, we only perform reduction steps on *innermost μ-replacing redexes*. Therefore, we have to restrict the definition of chains in order to obtain an appropriate notion corresponding to innermost *CSR*, which, obviously, is an adaptation of the one for standard *CSR*(see [AGL10]). Regarding innermost reductions, arguments of a redex should be in *normal form* before the redex is contracted and, regarding *CSR*, the redex to be contracted has to be in a *μ-replacing* position.

Given a signature $\mathcal{F}$ and $f \in \mathcal{F}$, we let $f^\sharp$ be a new fresh symbol (often called *tuple* symbol or DP-symbol) associated to a symbol f [AG00]. Let $\mathcal{F}^\sharp$ be the set of tuple symbols associated to symbols in $\mathcal{F}$. As usual, for $t = f(t_1,\ldots,t_k) \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we write $t^\sharp$ to denote the *marked* term $f^\sharp(t_1,\ldots,t_k)$. Conversely, given a marked term $t = f^\sharp(t_1,\ldots,t_k)$, where $t_1,\ldots,t_k \in \mathcal{T}(\mathcal{F},\mathcal{X})$, we write $t^\natural$ to denote the term $f(t_1,\ldots,t_k) \in \mathcal{T}(\mathcal{F},\mathcal{X})$. Let $\mathcal{T}^\sharp(\mathcal{F},\mathcal{X}) = \{t^\sharp \mid t \in \mathcal{T}(\mathcal{F},\mathcal{X}) - \mathcal{X}\}$ be the set of marked terms.

Definition 5 (Innermost Context-Sensitive Dependency Pairs) *Let $\mathcal{R} = (\mathcal{F},R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. We define* $\mathsf{iDP}(\mathcal{R},\mu) = \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu) \cup \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$ *to be the set of* innermost context-sensitive *dependency pairs (ICS-DPs) where:*

$$\begin{array}{lcl} \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu) & = & \{l^\sharp \to s^\sharp \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\trianglerighteq_\mu s, \mathrm{NARR}^\mu(\mathrm{REN}^\mu(s))\} \\ \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu) & = & \{l^\sharp \to x \mid l \to r \in R, l^\sharp \in \mathsf{NF}_\mu(\mathcal{R}), x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(l)\} \end{array}$$

We extend $\mu \in M_\mathcal{F}$ into $\mu^\sharp \in M_{\mathcal{F}\cup\mathcal{D}^\sharp}$ by $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, and $\mu^\sharp(f^\sharp) = \mu(f)$ if $f \in \mathcal{D}$.

Example 5 *Consider Example 1. The set* $\mathsf{iDP}(\mathcal{R},\mu)$ *consists of the following pairs:*

$$\begin{aligned}
\mathtt{MINUS}(\mathtt{s}(x),\mathtt{s}(y)) &\to \mathtt{MINUS}(x,y) && (1)\\
\mathtt{QUOT}(\mathtt{s}(x),s(y)) &\to \mathtt{MINUS}(x,y) && (2)\\
\mathtt{QUOT}(\mathtt{s}(x),s(y)) &\to \mathtt{QUOT}(\mathtt{minus}(x,y),\mathtt{s}(y)) && (3)\\
\mathtt{SEL}(\mathtt{s}(y),\mathtt{cons}(x,xs)) &\to \mathtt{SEL}(y,xs) && (4)\\
\mathtt{SEL}(\mathtt{s}(y),\mathtt{cons}(x,xs)) &\to xs && (5)\\
\mathtt{ZWQUOT}(\mathtt{cons}(x,xs),\mathtt{cons}(y,ys)) &\to \mathtt{QUOT}(x,y) && (6)
\end{aligned}$$

The ICSDPs $u \to v \in \mathsf{iDP}_{\mathcal{X}}(\mathcal{R},\mu)$ in Definition 5, consisting of collapsing rules only, are called the *collapsing* ICSDPs. A rule $l \to r$ of a TRS $\mathcal{R}$ is μ-conservative if $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$, i.e., it does not contain migrating variables; $\mathcal{R}$ is μ-conservative if all its rules are (see [Luc96, Luc06]).

Clearly, If $\mathcal{R}$ is μ-conservative, then $\mathsf{iDP}(\mathcal{R},\mu) = \mathsf{iDP}_{\mathcal{F}}(\mathcal{R},\mu)$.

Therefore, in order to deal with μ-conservative TRSs $\mathcal{R}$ we only need to consider the 'classical' dependency pairs in $\mathsf{iDP}_{\mathcal{F}}(\mathcal{R},\mu)$. If the TRS $\mathcal{R}$ contains non-μ-conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

To deal with the information corresponding to hidden terms and hiding contexts when trying to characterize innermost μ-termination with ICSDPs, we use an *unhiding TRS* $\mathsf{unh}(\mathcal{R},\mu)$. This unhiding TRS captures the situation described in Theorem 1 when managing migrating variables. According to this, we have to remove the (instance of the) hiding context $C_i[]$ to extract the delayed call t_i and then connect this delayed call, which is an instance $\theta(t'_i)$ of a hidden term t'_i with the next pair in the innermost μ-chain. We perform these two actions by using two kind of rewrite rules:

- If $\theta(Ci[t'_i]) = \theta(f(t_1,\ldots,t_{i-1},C'_i[t'_i],t_{i+1},\ldots,t_k))$ then, since $C_i[]$ is a hiding context, f hides position i and $C'_i[]$ is a hiding context as well. Then, we can extract $\theta(C'_i[t'_i])$ from $\theta(C_i[t'_i])$ by using the following projection rule: $f(x_1,\ldots,x_{i-1},x_i,x_{i+1},\ldots,x_k) \to x_i$
- Once t_i has been reached, we know that it is an instance $t_i = \theta(t'_i)$ of a nonvariable hidden term $t'_i \in \mathcal{NHT}(\mathcal{R},\mu)$ and we have to connect t_i with the next innermost context-sensitive dependency pair. Since the root of the innermost context-sensitive dependency pair is a marked symbol, we can do it by using a rule that just changes the root symbol by its marked version in the following way: $t'_i \to t'^{\sharp}_i$

Definition 6 (Unhiding TRS [GL10]) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We define* $\mathsf{unh}(\mathcal{R},\mu)$ *as the TRS consisting of the following rules:*

1. $f(x_1,\ldots,x_i,\ldots,x_k) \to x_i$ *for all function symbols f of arity k, distinct variables $x_1,\ldots,x_k$, and $1 \le i \le k$ such that f* hides position *i in $l \to r \in R$, and*

2. $t \to t^\sharp$ *for every* $t \in \mathcal{NHT}(\mathcal{R},\mu)$.

Example 6 *The unhiding TRS* $\mathsf{unh}(\mathcal{R},\mu)$ *for* $\mathcal{R}$ *and* μ *in Example 1 is:*

$$\begin{array}{rcl} \mathtt{from}(x) & \to & \mathtt{FROM}(x) \\ \mathtt{zWquot}(x,y) & \to & \mathtt{ZWQUOT}(x,y) \\ \mathtt{zWquot}(x,y) & \to & x \\ \mathtt{zWquot}(x,y) & \to & y \end{array}$$

Definitions 5 and 6 lead to a suitable notion of *chain* which captures minimal infinite μ-rewrite sequences according to the description in Theorem 1. In the following, given a TRS $\mathcal{S}$, we let $\mathcal{S}_{\rhd_\mu}$ be the rules from $\mathcal{S}$ of the form $s \to t \in \mathcal{S}$ and $s \rhd_\mu t$ (Definition 6-1); and $\mathcal{S}_\sharp = \mathcal{S} \setminus \mathcal{S}_{\rhd_\mu}$(Definition 6-2).

As in the DP-framework [GTS04, GTSF06], where the procedence of *pairs* does not matter, we rather think of another TRS $\mathcal{P}$ which is used together with $\mathcal{R}$ to build the chains. Once this more abstract notion of chain is introduced, it can be particularized to be used with ICSDPs, by just taking $\mathcal{P} = \mathsf{iDP}(\mathcal{R},\mu)$.

Definition 7 ((Minimal) Innermost μ-Chain) *Let* $\mathcal{R}$*,* $\mathcal{P}$ *and* $\mathcal{S}$ *be TRSs and* $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$*. An innermost* $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$*-chain is a finite or infinite sequence of pairs* $u_i \to v_i \in \mathcal{P}$*, together with a substitution* σ *satisfying that, for all* $i \geq 1$*,* $\sigma(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$ *and :*

1. *if* $v_i \notin \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^\mu(u_i)$*, then* $\sigma(v_i) = t_i \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathrm{i}} \sigma(u_{i+1})$*, and*
2. *if* $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^\mu(u_i)$*, then* $\sigma(v_i) \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{S}_{\rhd_\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} t_i \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathrm{i}} \sigma(u_{i+1})$.

An innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$*-chain is called* minimal *if for all* $i \geq 1$*,* t_i *is innermost* $(\mathcal{R},\mu)$*-terminating.*

Note that the condition $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^\mu(u_i)$ in Definition 8 implies that v_i is a variable. Furthermore, since each $u_i \to v_i \in \mathcal{P}$ is a rewrite rule (i.e., $\mathcal{V}ar(v_i) \subseteq \mathcal{V}ar(u_i)$), v_i is a *migrating variable* in the rule $u_i \to v_i$.

In the following, the pairs in a CS-TRS $(\mathcal{P},\mu)$, where $\mathcal{P} = (\mathcal{G},P)$, are partitioned according to its role in Definition 8 as follows:

$$P_\mathcal{X} = \{u \to v \in P \mid v \in \mathcal{V}ar(u) \setminus \mathcal{V}ar^\mu(u)\} \text{ and } P_\mathcal{G} = P - P_\mathcal{X}$$

Despite this fact, we refer to $\mathcal{P}_\mathcal{X}$ as the set of *collapsing* pairs in $\mathcal{P}$ because its intended role in Definition 8 is capturing the computational behavior of collapsing ICSDPs in $\mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$.

The following result establishes the soundness of the innermost context-sensitive dependency pair approach. As usual, in order to fit the requirement of variable-disjointness among two arbitrary pairs in a chain of pairs, we assume that appropriately renamed ICSDPs are available when necessary.

Theorem 2 (Soundness) *Let* $\mathcal{R}$ *be a TRS and* $\mu \in M_\mathcal{R}$*. If there is no infinite minimal innermost* $(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp,\mathbf{i})$*-chain, then* $\mathcal{R}$ *is innermost* μ*-terminating.*

PROOF.

By contradiction. If $\mathcal{R}$ is not innermost μ-terminating, then by Proposition 2 (2) there is $t \in \mathcal{T}_{\infty,\mu,\mathsf{i}}$. By Theorem 1, there are rules $l_i \to r_i \in \mathcal{R}$, matching substitutions σ_i, such that $\sigma_i(l_i)$ is argument μ-normalized and terms $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, for $i \geq 1$ such that

$$t = t_0 \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^{*} \sigma_1(l_1) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_1(r_1) \trianglerighteq_\mu t_1 \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^{*} \sigma_2(l_2) \overset{\Lambda}{\hookrightarrow}_{\mathsf{i}} \sigma_2(r_2) \trianglerighteq_\mu t_2 \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^{*} \cdots$$

where either (D1) $t_i = \sigma_i(s_i)$ for some s_i such that $r_i \trianglerighteq_\mu s_i$ or (D2) $\sigma_i(x_i) = C_i[t_i]$ for some $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(\ell_i)$ and $C_i[t_i] = \theta_i(C_i'[t_i'])$ for some $t_i' \in \mathcal{NHT}$ and hiding context $C_i'[\Box]$. Furthermore, since $t_{i-1} \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^{*} \sigma_i(l_i)$ and $t_{i-1} \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ (in particular, $t_0 = t \in \mathcal{T}_{\infty,\mu,\mathsf{i}} \subseteq \mathcal{M}_{\infty,\mu,\mathsf{i}}$), by Proposition 2 (3), $\sigma_i(l_i) \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$ for all $i \geq 1$. Note that, since $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, we have that $t_i^\sharp$ is innermost μ-terminating (with respect to $\mathcal{R}$), because all μ-replacing subterms of t_i (hence of $t_i^\sharp$ as well) are innermost μ-terminating and $root(t^\sharp)$ is not a defined symbol of $\mathcal{R}$.

First, note that $\mathsf{iDP}(\mathcal{R},\mu)$ is a TRS $\mathcal{P}$ over the signature $\mathcal{G} = \mathcal{F} \cup \mathcal{D}^\sharp$ and $\mu^\sharp \in M_{\mathcal{F}\cup\mathcal{G}}$ as required by Definition 8. Furthermore, $\mathcal{P}_\mathcal{G} = \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu)$ and $\mathcal{P}_\mathcal{X} = \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$. We can define an infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathtt{i})$-chain using ICSDPs $u_i \to v_i$ for $i \geq 1$, where $u_i = l_i^\sharp$, and

1. $v_i = s_i^\sharp$ if (D1) holds. Since $t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$, we have that $root(s_i) \in \mathcal{D}$ and, because $t_i = \sigma_i(s_i)$ and $\sigma_i(s_i) \hookrightarrow_{\mathsf{i}}^{*} \sigma_{i+1}(\ell_{i+1})$, by Corollary 2 $\textsc{Ren}^\mu(s_i)$ is μ-narrowable. Furthermore, if we assume that s_i is a μ-replacing subterm of l_i (i.e., $l_i \rhd_\mu s_i$), then $\sigma_i(l_i) \rhd_\mu \sigma_i(s_i)$ which (since $\sigma_i(s_i) = t_i \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$) contradicts that $\sigma_i(l_i) \in \mathcal{M}_{\infty,\mu,\mathsf{i}}$. Thus, $l_i \not\rhd_\mu s_i$. Moreover, since $\sigma_i(l_i)$ is argument μ-normalized, it implies that $\sigma_i(l_i^\sharp)$ also, which means that $\sigma_i(l_i^\sharp) \in \mathsf{NF}_\mu(\mathcal{R})$ (since $root(l_i^\sharp)$ is not a defined symbol of $\mathcal{R}$) and trivially also is $l_i^\sharp$. Hence, $u_i \to v_i \in \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu)$. Furthermore, by minimality, $t_i^\sharp = \sigma_i(v_i)$ is innermost μ-terminating. Finally, since $t_i = \sigma_i(s_i) \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}}^{*} \sigma_{i+1}(l_{i+1})$ and $\mu^\sharp$ extends μ to $\mathcal{F} \cup \mathcal{D}^\sharp$ by $\mu^\sharp(f^\sharp) = \mu(f)$ for all $f \in \mathcal{D}$, we also have that $\sigma_i(v_i) = \sigma_i(s_i^\sharp) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma_{i+1}(u_{i+1})$.

2. $v_i = x_i$ if (D2) holds. Again, since $\sigma_i(l_i)$ is argument μ-normalized, it implies that $\sigma_i(l_i^\sharp)$ also, which means that $\sigma_i(l_i^\sharp) \in \mathsf{NF}_\mu(\mathcal{R})$ (since $root(l_i^\sharp)$ is not a defined symbol of $\mathcal{R}$) and trivially also is $l_i^\sharp$. Clearly, $u_i \to v_i \in \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$. As discussed above, $t_i^\sharp$ is innermost μ-terminating by minimality. Since $\sigma_i(x_i) = C_i[t_i]$, we have that $\sigma_i(v_i) = C_i[t_i]$. By the hiding property, we know that $C_i[\,]$ is an instance of hiding context $C_i'[\,]$, then we have that $\theta_i(C_i')[t_i] \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_{\rhd_\mu},\mu}^{*} t_i$. And we also know that t_i is an instance $\theta_i(t_i')$ of a hidden term $t_i' \in \mathcal{NHT}(\mathcal{R},\mu)$. Thus $t_i' \to t_i'^\sharp \in \mathcal{S}_\sharp$ and

we have $\theta(t'_i) \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} \theta(t_i'^\sharp)$. Finally, since $t_i \overset{>\Lambda}{\longrightarrow}_{\mathsf{i}}^* \sigma_{i+1}(l_{i+1})$, again we have that $t_i^\sharp \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma_{i+1}(u_{i+1})$.

Regarding σ, w.l.o.g. we can assume that $\mathcal{V}ar(l_i) \cap \mathcal{V}ar(l_j) = \varnothing$ for all $i \neq j$, and therefore $\mathcal{V}ar(u_i) \cap \mathcal{V}ar(u_j) = \varnothing$ as well. Then, σ is given by $\sigma(x) = \sigma_i(x)$ whenever $x \in \mathcal{V}ar(u_i)$ for $i \geq 1$. From the discussion in points (1) and (2) above, we conclude that the ICSDPs $u_i \to v_i$ for $i \geq 1$ together with σ define an infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$-chain which contradicts our initial assumption. □

Now we prove that the use of ICSDPs is not only correct but also complete for proving innermost termination of *CSR*.

Theorem 3 (Completeness) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. If $\mathcal{R}$ is innermost μ-terminating, then there is no infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$-chain.*

Proof.

By contradiction. If there is an infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$-chain, then there is a substitution σ and dependency pairs $u_i \to v_i \in \mathsf{iDP}(\mathcal{R},\mu)$ such that $\sigma(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$ and

1. $\sigma(v_i) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma(u_{i+1})$, if $u_i \to v_i \in \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu)$, and
2. if $u_i \to v_i = u_i \to x_i \in \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$, then $\sigma(v_i) \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{S}_{\rhd\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} t_i \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_{i+1})$.

for $i \geq 1$. Now, consider the first dependency pair $u_1 \to v_1$ in the sequence:

1. If $u_1 \to v_1 \in \mathsf{iDP}_\mathcal{F}(\mathcal{R},\mu)$, then $v_1^\natural$ is a μ-replacing subterm of the right-hand-side r_1 of a rule $l_1 \to r_1$ in $\mathcal{R}$. Therefore, $r_1 = C_1[v_1^\natural]_{p_1}$ for some $p_1 \in \mathcal{P}os^\mu(r_1)$ and, since $\sigma(u_1) \in \mathsf{NF}_\mu(\mathcal{R})$, we can perform the innermost μ-rewriting step $s_1 = \sigma(u_1^\natural) \hookrightarrow_{\mathcal{R},\mu,\mathsf{i}} \sigma(r_1) = \sigma(C_1)[\sigma(v_1^\natural)]_{p_1} = t_1$, where $\sigma(v_1^\natural)^\sharp = \sigma(v_1) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma(u_2)$ and $\sigma(u_2)$ also initiates an infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu), \mathcal{R}, \mathsf{unh}(\mathcal{R},\mu), \mu^\sharp, \mathbf{i})$-chain. Note that $p_1 \in \mathcal{P}os^\mu(t_1)$.
2. If $u_1 \to x_1 \in \mathsf{iDP}_\mathcal{X}(\mathcal{R},\mu)$, then there is a rule $l_1 \to r_1$ in $\mathcal{R}$ such that $u_1 = l_1^\sharp$, and $x_1 \in \mathcal{V}ar^\mu(r_1) \setminus \mathcal{V}ar^\mu(l_1)$, i.e., $r_1 = C_1[x_1]_{q_1}$ for some $q_1 \in \mathcal{P}os^\mu(r_1)$. Furthermore, if $\sigma(v_1) = \sigma(x_1) = C_1[t_1] \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{S}_{\rhd\mu},\mu} t_1$ this means that $C_1[\Box]$ is an instance of a hiding context $C'_1[\Box]$, $C_1[\Box] = \theta_1(C'_1)[\Box]$. Furthermore, t_1 is μ-replacing in $C_1[t_1]$. If $t_1 \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} t_1^\sharp$ means that $t_1 = \theta_1(t'_1)$ for some $t'_1 \in \mathcal{DHT}$, then since $\sigma(u_1) = \sigma(l_1)^\sharp \in \mathsf{NF}_\mu(\mathcal{R})$, we can perform the innermost μ-rewriting step $s_1 = \sigma(l_1) \hookrightarrow_{\mathcal{R},\mu,\mathsf{i}} \sigma(r_1) = \sigma(C_1)[C'_1[t_1]_{p'_1}]_{q_1} = s_1$ where $t_1^\sharp \overset{!}{\hookrightarrow}_{\mathcal{R},\mu^\sharp,\mathsf{i}} \sigma(u_2)$ (hence $t_1 \overset{!}{\hookrightarrow}_{\mathsf{i}} u_2^\natural$) and $\sigma(u_2)$

initiates an infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp,\mathbf{i})$-chain. Note that $p_1 = q_1.p_1' \in \mathcal{P}os^\mu(s_1)$ where p_1' is the position of the hole in $C_1[\Box]_{p_1'}$.

Since $\mu^\sharp(f^\sharp) = \mu(f)$, and $p_1 \in \mathcal{P}os^\mu(s_1)$, we have that $t_1 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathrm{i}} s_2[\sigma(u_2)]_{p_1} = s_2$ and $p_1 \in \mathcal{P}os^\mu(s_2)$. Therefore, we can build in that way an infinite innermost μ-rewrite sequence

$$s_1 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} t_1 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathrm{i}} s_2 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} \cdots$$

which contradicts the innermost μ-termination of $\mathcal{R}$.

□

As a corollary of Theorems 2 and 3, we have.

Corollary 3 (Characterization of innermost μ-termination) *Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$. Then, $\mathcal{R}$ is innermost μ-terminating if and only if there is no infinite minimal innermost $(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp,\mathbf{i})$-chain.*

Example 7 *Consider the following TRS $\mathcal{R}$:*

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c}(\mathtt{b}) \\ \mathtt{f}(\mathtt{c}(x),x) & \to & \mathtt{f}(x,x) \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ and $\mu(\mathtt{c}) = \varnothing$. There is only one ICSDP:

$$\mathtt{F}(\mathtt{c}(x),x) \to \mathtt{F}(x,x)$$

Since $\mu^\sharp(\mathtt{F}) = \{1,2\}$, if a substitution σ satisfies $\sigma(\mathtt{F}(\mathtt{c}(x),x)) \in \mathsf{NF}_\mu(\mathcal{R})$, then $\sigma(x) = s$ is in μ-normal form. Assume that the dependency pair is part of an innermost μ-chain. Since there is no way to μ-rewrite $F(s,s)$, there must be $F(s,s) = F(c(t),t)$ for some term t, which means that $s = t$ and $c(t) = s$, i.e., $t = c(t)$ which is not possible. Thus, there is no infinite innermost chain of ICSDPs for $\mathcal{R}$, which is proved innermost terminating by Theorem 2.

Of course, ad-hoc reasonings like in Example 7 do not lead to automation. In following sections we discuss how to prove termination of innermost *CSR* by giving *constraints* on terms that can be solved by using standard methods.

5 Mechanizing Proofs of Innermost μ-termination

Regarding termination proofs, the central notion in the Dependency Pair Framework [GTS04, GTSF06, Thi07] is that of *DP-termination problem*: given a TRS $\mathcal{R}$ and a set of pairs $\mathcal{P}$, the goal is checking the absence (or presence) of infinite (minimal) chains. Termination of a TRS $\mathcal{R}$ is addressed as a DP-termination

problem where $\mathcal{P} = \mathsf{DP}(\mathcal{R})$. The most important notion regarding mechanization of the proofs is that of *processor*. A (correct) processor basically transforms DP-termination problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original DP-termination problem implies the existence of an infinite chain in the transformed one. Here 'simpler' usually means that fewer pairs are involved. Often, processors are not only correct but also *complete*, i.e., there is an infinite minimal chain in the original DP-termination problem if and only if there is an infinite minimal chain in the transformed problem. This is essential if we are interested in *disproving* termination.

In [AEF+08, AGL10, GL10], we have developed a CSDP framework for *CSR*. In this chapter, we extend the CSDP framework developed in [GL10] to innermost *CSR*. First, we recall the definition of chain for standard *CSR*.

Definition 8 ((Minimal) μ-Chain of Pairs [GL10]) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. A $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution σ satisfying that, for all $i \geq 1$,*

1. *if $v_i \notin \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$, then $\sigma(v_i) = t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$, and*
2. *if $v_i \in \mathcal{V}ar(u_i) \setminus \mathcal{V}ar^{\mu}(u_i)$, then $\sigma(v_i) \xhookrightarrow{\Lambda}{}^*_{\mathcal{S}_{\rhd\mu},\mu} \circ \xhookrightarrow{\Lambda}_{\mathcal{S}_\sharp,\mu} t_i \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_{i+1})$.*

A $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain is called minimal *if for all $i \geq 1$, t_i is $(\mathcal{R},\mu)$-terminating.*

Definition 9 (CS Problem) *A CS problem τ is a tuple $\tau = (\mathcal{P},\mathcal{R},\mathcal{S},\mu,e)$, where $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ are TRSs, and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$ and $e \in \{\mathbf{t},\mathbf{i}\}$ is a flag that stands for* **t**ermination *or* **i**nnermost termination *of* CSR. *The* CS problem $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,e)$ *is* finite *if there is no infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,e)$-chain. The* CS problem $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,e)$ *is* infinite *if $\mathcal{R}$ is non-μ-terminating (for $e = \mathbf{t}$) or innermost non-μ-terminating (for $e = \mathbf{i}$) or there is an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,e)$-chain.*

Definition 10 (CS Processor) *A CS processor* Proc *is a mapping from CS problems into sets of CS problems. Alternatively, it can also return* "no"*. A CS processor* Proc *is*

- sound *if for all CS problems τ, τ is finite whenever $\mathsf{Proc}(\tau) \neq \mathsf{no}$ and $\forall \tau' \in \mathsf{Proc}(\tau)$, τ' is finite.*
- complete *if for all CS problems τ, τ is infinite whenever $\mathsf{Proc}(\tau) = \mathsf{no}$ or $\exists \tau' \in \mathsf{Proc}(\tau)$ such that τ' is infinite.*

Now we have the following result which extends the framework in [GL10] to innermost *CSR*.

Theorem 4 (CSDP Framework) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We construct a tree whose nodes are labeled with CS problems or* "yes" *or* "no"*, and whose root is labeled with $(\mathcal{P},\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp,e)$, where $\mathcal{P} = \mathsf{DP}(\mathcal{R},\mu)$ if $e = \mathbf{t}$ and $\mathcal{P} = \mathsf{iDP}(\mathcal{R},\mu)$ if $e = \mathbf{i}$. For every inner node labeled with τ, there is a sound processor* Proc *satisfying one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child, labeled with* "no".
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child, labeled with* "yes".
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$, $\mathsf{Proc}(\tau) \neq \varnothing$, *and the children of the node are labeled with the CS problems in* $\mathsf{Proc}(\tau)$.

If all leaves of the tree are labeled with "yes", *then* $\mathcal{R}$ *is innermost* μ*-terminating. Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then* $\mathcal{R}$ *is not innermost* μ*-terminating.*

In following sections we describe a number of sound and (most of them) complete CS-processors for proving termination of innermost *CSR*. First, we formalize some basic processors.

6 CS Basic Processors for Innermost Termination of *CSR*

In standard rewriting, Gramlich, showed that termination and innermost termination coincide for locally confluent overlay TRSs $\mathcal{R}$[Gra95, Theorem 3.23]. Thus, his result allows us to prove termination of such TRSs $\mathcal{R}$by proving innermost termination of $\mathcal{R}$. Although local confluence is undecidable, every nonoverlapping rewrite system is also a locally confluent overlay system, therefore, this approximation is commonly adopted. However, for context-sensitive rewriting this is not enough. This fact was noticed by Lucas in a personal communication showing the following example:

Example 8 *Consider the following TRS* $\mathcal{R}$*:*

$$\begin{array}{rcl} \mathtt{f}(x,x) & \to & \mathtt{b} \\ \mathtt{f}(x,g(x)) & \to & \mathtt{f}(x,x) \\ \mathtt{c} & \to & \mathtt{g}(c) \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ *and* $\mu(\mathtt{g}) = \varnothing$*. This system is nonoverlapping and innermost* μ*-terminating, but not* μ*-terminating since* $\mathtt{f}(\mathtt{c},\mathtt{c}) \hookrightarrow_\mu \underline{\mathtt{f}(\mathtt{c},\mathtt{g}(\mathtt{c}))} \hookrightarrow_\mu \mathtt{f}(\mathtt{c},\mathtt{c}) \hookrightarrow_\mu \cdots$

Later, in [GL02b, GM02b] it is proved that the equivalence between termination of innermost *CSR* and termination of *CSR* holds in some interesting cases. Thanks to this, the following result was formulated:

Theorem 5 *[GM02b] Let* $\mathcal{R} = (\Sigma, R)$ *be an orthogonal TRS and* $\mu \in M_\Sigma$*.* $\mathcal{R}$ *is* μ*-terminating if and only if it is innermost* μ*-terminating.*

A similar result can be found in [GL02b]. First, we need the following definition:

Definition 11 *[Luc98, Definition 5] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$. $\mathcal{R}$ has* left-homogeneous replacing variables *(LHRV for short) if, for every μ-replacing variable x in the left-hand side l of a rule $l \to r \in R$, all occurrences of x are replacing in both, l and r.*

Theorem 6 *[GL02b, Theorem 7] Let $\mathcal{R} = (\Sigma, R)$ be a TRS and $\mu \in M_{\mathcal{R}}$ be such that $\mathcal{R}$ is a locally confluent overlay system satisfying LHRV. If $\mathcal{R}$ is innermost μ-terminating, then it is also μ-terminating.*

So, whenever it is possible, we switch to innermost μ-termination since proofs are often easier due to the fact that when considering an innermost rewriting step, we know that every possible subterm of our redex is in normal form with respect to our rewriting relation. For instance, this is shown when estimating the graph.

On the other hand, we have developed a huge amount of processors for proving termination of *CSR* [AGL10, GL10] and it is also interesting to use them in proofs of innermost termination of *CSR*.

Theorem 7 (Commuting Processors) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem such that*

1. *$(\mathcal{R} \cup \mathcal{P} \cup \mathcal{S})$ is nonoverlapping and satisfies LHRV, or*
2. *$(\mathcal{R} \cup \mathcal{P} \cup \mathcal{S})$ is orthogonal*

Then, the processors $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$ and $\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}$ given by

$$\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t}) = \begin{cases} \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{if (1) or (2)} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{otherwise} \end{cases}$$

$$\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})\} & \text{if (1) or (2)} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

are sound and $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$ also complete.

Proof. Regarding soundness of $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{t})$-chain A, but there is no infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain. Due to the finiteness of $\mathcal{P}$ and $\mathcal{S}$, we can assume that there are subsets $\mathcal{Q} \subseteq \mathcal{P}$ and $\mathcal{T} \subseteq \mathcal{S}$ such that A has a tail B

$$\sigma(u_1)\left\{ \begin{array}{l} \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} \\ \hookrightarrow_{\mathcal{Q}_\mathcal{X},\mu} \circ \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{T}_{\triangleright\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{T}_\sharp,\mu} \end{array} \right\} t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)\left\{ \begin{array}{l} \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} \\ \hookrightarrow_{\mathcal{Q}_\mathcal{X},\mu} \circ \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{T}_{\triangleright\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{T}_\sharp,\mu} \end{array} \right\} t_2 \hookrightarrow^*_{\mathcal{R},\mu} \cdots$$

for some substitution σ, where all pairs in $\mathcal{Q}$ and all rules in $\mathcal{T}$ are infinitely often used (note that, if $\mathcal{T} \neq \varnothing$, then $\mathcal{T}_\sharp \neq \varnothing$ and $\mathcal{Q}_\mathcal{X} \neq \varnothing$), and for all $i \geq 1$, (a) if $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$, then $t_i = \sigma(v_i)$ and (b) if $u_i \to v_i = u_i \to x_i \in \mathcal{Q}_\mathcal{X}$, then $\sigma(u_i) \hookrightarrow_{\mathcal{Q}_\mathcal{X}} \circ \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{T}_{\triangleright\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{T}_\sharp,\mu} t_i$. Moreover, all t_i are $(\mathcal{R}, \mu)$-terminating. W.l.o.g. we can assume that $\sigma(u_1)$ is $(\mathcal{R}, \mu)$-terminating.

Proof of item (1) follows [GL02b] result for *CSR* and the details can be found in the original paper for equivalence between innermost termination and termination of TRSs [Gra95]. Proof of item (2) follows the results in [GM02b] and more precisely, the ones in [Emm08] that adapt them to the CSDP framework of [AEF+08]. For details about some statements in the following, we recall the reader to the corresponding lemmas in these papers to simplify the proof. Both shares the main idea of [Gra95] about using a transformation Φ which (*uniquely* since in both cases the system is nonoverlaping) μ-normalizes all maximal subterms of a given term with respect to $\mathcal{R}$ (therefore, top parts of the pairs are untouched). Formally,

$$\Phi_\mu(t) = C[{t_1}^{\mathsf{i}}{\downarrow_{(\mathcal{R},\mu)}}, \ldots, {t_n}^{\mathsf{i}}{\downarrow_{(\mathcal{R},\mu)}}]$$

where $t = C[t_1, \ldots, t_n]$ and $t_1, \ldots, t_n$ are innermost μ-terminating subterms at active positions. Clearly $t \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} \Phi_\mu(t)$. The main difference dealing with context-sensitive rewriting arises in the synchronization within variable parts since e.g. one occurrence of a variable x can be active while another can be inactive. This is solved requiring linearity of left-hand sides (en the case of condition (2)) or $LHRV$ (in the case of condition (1)).

Let $u'_1 = \Phi_\mu(\sigma(u_1))$, therefore, u'_1 is an innermost $(\mathcal{R},\mu)$-terminating instance of u_1 and there exists a substitution σ' s.t.

$$\sigma(u_1) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} u'_1 = \sigma'(u_1) \left\{ \begin{array}{c} \overset{}{\hookrightarrow}_{\mathcal{Q}_\mathcal{G},\mu} \\ \hookrightarrow_{\mathcal{Q}_\mathcal{X},\mu} \circ \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{T}_{\rhd_\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{T}_\sharp,\mu} \end{array} \right\} \sigma'(t_1)$$

and $\sigma'(t_1) = \Phi_\mu(\sigma(t_1))$ is innermost $(\mathcal{R},\mu)$-terminating. Paying attention in the part $t_1 \hookrightarrow^*_{\mathcal{R},\mu} \sigma(u_2)$, since by minimality, t_i are $(\mathcal{R},\mu)$-terminating, all contracted redexes in the sequence also will be. Therefore, we can reach the point where $\sigma'(t_1) \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} u'_2$ such that $u'_2 = \Phi_\mu(\sigma(u_2))$ and u'_2 is innermost $(\mathcal{R},\mu)$-terminating.

Since for all pair $u_i \to v_i$ asume variable disjoint, the new substitution can be extended to $\sigma'(u_2) = u'_2$. Reasoning in this way, the original infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain can be seen as:

$$u'_1 \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} \sigma'(u_1) \left\{ \begin{array}{c} \overset{}{\hookrightarrow}_{\mathcal{Q}_\mathcal{G},\mu} \\ \hookrightarrow_{\mathcal{Q}_\mathcal{X},\mu} \circ \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{T}_{\rhd_\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{T}_\sharp,\mu} \end{array} \right\} \sigma'(t_1) \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} u'_2 \hookrightarrow^*_{\mathcal{R},\mu,\mathsf{i}} \sigma'(u_2) \cdots$$

where $\sigma'(t_i)$ is innermost $(\mathcal{R},\mu)$-terminating and $\sigma'(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$ for all $i \geq 1$. Therefore we get an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain, leading to a contradiction.

Regarding completeness of $\mathsf{Proc}_{\mathbf{t}\to\mathbf{i}}$, since if $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$ is infinite, that means that $\mathcal{R}$ is not innermost μ-terminating and therefore $\mathcal{R}$ is not μ-terminating or there is an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain and, since condition (1) or (2) hold then the equivalence between innermost μ-termination and μ-termination comes from Theorems 5 and 6 respectively and t_i is $(\mathcal{R},\mu)$-terminating and therefore there is also an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain. Therefore $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$ is infinite.

We prove soundness of $\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}$ by contradiction. Assume that there is an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain but there is no infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$. Since condition (1) or (2) hold, reasoning as above every minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain is also a minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain, therefore there is an infinite minimal $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{t})$-chain, leading to a contradiction.

□

Soundness of $\mathsf{Proc}_{\mathbf{i}\to\mathbf{t}}$ needs to impose the requirements about equivalence between innermost μ-termination and μ-termination since we are dealing with minimal chains. Obviously, it is always possible to prove innermost μ-termination of a TRS by proving μ-termination without taking into account any additional condition but this cannot be done when managing minimality.

The following proposition establishes some important 'basic' cases of (absence of) infinite context-sensitive chains of pairs which are used later and with slight differences were presented in [AGL10]. Note that in the innermost case they also hold.

Proposition 5 *Let $\mathcal{R}=(\mathcal{F},R)$ and $\mathcal{P}=(\mathcal{G},P)$ and $\mathcal{S}=(\mathcal{H},S)$ be TRSs and $\mu\in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$.*

1. *If $P=\varnothing$, then there is no $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*
2. *If $R=\varnothing$, then there is no infinite $(\mathcal{P}_\mathcal{X},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*
3. *Let $u\to v\in\mathcal{P}_\mathcal{G}$ be such that $v'=\theta(u)$ for some substitution θ such that $\theta(u)\in\mathsf{NF}_\mu(\mathcal{R})$ and renamed version v' of v. Then, there is an infinite innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain.*

Proof.

1. Trivial.
2. By contradiction. If there is an infinite $(\mathcal{P}_\mathcal{X},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain, then, since there is no rule in $\mathcal{R}$, there is a substitution σ such that

$$\sigma(u_1)\hookrightarrow_{\mathcal{P},\mu}\sigma(x_1)\xleftarrow{\Lambda}{}^{*}_{\mathcal{S}_{\rhd_\mu},\mu}s_1\overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu}t_1=\sigma(u_2)\hookrightarrow_{\mathcal{P},\mu}\sigma(x_2)\xleftarrow{\Lambda}{}^{*}_{\mathcal{S}_{\rhd_\mu},\mu}s_2\overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu}\cdots$$

   For $i\geq 1$, since $x_i\in\mathcal{V}ar(u_i)$ and u_i is not a variable, we have $u_i\rhd x_i$, hence $\sigma(u_i)\rhd\sigma(x_i)$ (by stability of $\rhd$), and also $\sigma(u_i)\rhd s_i$. Since s_i and $\sigma(u_{i+1})$ only differ in the root symbol, we can actually say that $s_i\rhd s_{i+1}$ for all $i\geq 1$. Thus, we obtain an infinite sequence $s_1\rhd s_2\rhd\cdots$ which contradicts the well-foundedness of $\rhd$.

3. Since we always deal with *renamed* versions $u_i\to v_i$ of the pair $u\to v\in\mathcal{P}$, for each $x\in\mathcal{V}ar(u)$, we write x_i to denote the 'name' of the variable x in $u_i\to v_i$. According to our hypothesis, we can assume the existence of substitutions θ_{i+1} such that $v_i=\theta_{i+1}(u_{i+1})$. Note that, for all $x\in\mathcal{V}ar(u)$ and $i\geq 1$, $\mathcal{V}ar(\theta_{i+1}(u_{i+1}))\subseteq\mathcal{V}ar(v_i)\subseteq\mathcal{V}ar(u_i)$ and $\theta(u)\in\mathsf{NF}_\mu(\mathcal{R})$ is needed to deal only with innermost μ-chains.

We can define an infinite innermost $(\{u \to v\}, \varnothing, \varnothing, \mu, \mathbf{i})$-chain (hence an innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain) by using the renamed versions $u_i \to v_i$ of $u \to v$ for $i \geq 1$ together with σ given (inductively) as follows: for all $x \in \mathcal{V}ar(u)$, $\sigma(x_1) = x_1$ and $\sigma(x_i) = \sigma(\theta_i(x_i))$ for all $i > 1$. Note that $\sigma(v_i) = \sigma(\theta_{i+1}(u_{i+1})) = \sigma(u_{i+1})$ for all $i \geq 1$.

□

According to Proposition 5, for some specific CS problems it is easy to say whether they are finite or not.

Theorem 8 (Basic Innermost CS Processors) *Let $\mathcal{R} = (\mathcal{F}, R)$, $\mathcal{P} = (\mathcal{G}, P)$ and $\mathcal{S} = (\mathcal{H}, S)$ be TRSs and $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$.*

Then, the processors Proc_{Fin} and Proc_{Inf} given by

$$\mathsf{Proc}_{Fin}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \varnothing & \text{if } P = \varnothing \vee (R = \varnothing \wedge \mathcal{P} = \mathcal{P}_{\mathcal{X}}); \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

$$\mathsf{Proc}_{Inf}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \mathsf{no} & \text{if } v = \theta(u) \text{ and } \theta(u) \in \mathsf{NF}_{\mu}(\mathcal{R}) \\ & \text{for some } u \to v \in \mathcal{P}_{\mathcal{G}} \text{ and substitution } \theta; \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

are sound and complete.

The CS problems in Theorem 8 provide the necessary *base* cases for our proofs of innermost termination of *CSR*.

In the following sections we are going to show some powerful techniques adapted from standard rewriting to deal with proofs of innermost termination of *CSR*.

7 Innermost Context-Sensitive Dependency Graph

The analysis of infinite (minimal) chains of pairs is essential in the (CS)DP framework [GTSF06, AGL10, GL10]. Following [GL10], for innermost *CSR* we have the following.

Definition 12 (Innermost Context-Sensitive Graph of Pairs) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$. The* innermost context-sensitive (ICS) graph *$\mathsf{IG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ has $\mathcal{P}$ as the set of nodes. Given $u \to v, u' \to v' \in \mathcal{P}$, there is an arc from $u \to v$ to $u' \to v'$ if $u \to v, u' \to v'$ is a minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain for some substitution σ.*

In termination proofs, we are concerned with the analysis of *strongly connected components* (SCCs). A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle which is not contained in any other cycle. The following result justifies the use of SCCs for proving the absence of infinite minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chains.

Theorem 9 (SCC processor [GL10]) *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. Then, the processor* Proc_{SCC} *given by*

$\mathsf{Proc}_{SCC}(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i}) = \{(\mathcal{Q},\mathcal{R},\mathcal{S}_\mathcal{Q},\mu,\mathbf{i}) \mid \mathcal{Q}$ *contains the pairs of an SCC in* $\mathsf{IG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)\}$

(where $\mathcal{S}_\mathcal{Q}$ are the rules from $\mathcal{S}$ involving a possible $(\mathcal{Q},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain) is sound and complete.

As a consequence of this theorem, we can *separately* work with the strongly connected components of $\mathsf{IG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$, disregarding other parts of the graph.

Now we can use these notions to introduce the innermost context-sensitive dependency graph, i.e., the graph whose nodes instead of being arbitrary pairs are the ICSDPs ($\mathcal{P} = \mathsf{iDP}(\mathcal{R},\mu)$).

Definition 13 (Innermost Context-Sensitive Dependency Graph (ICS-DG)) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. The* Innermost Context-Sensitive Dependency Graph *associated to $\mathcal{R}$ and μ is* $\mathsf{IDG}(\mathcal{R},\mu) = \mathsf{IG}(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$.

7.1 Estimating the ICS Graph

In general, the innermost context-sensitive graph of a CS problem is *not* computable: it involves reachability of $\theta'(u')$ from $\theta(v)$ (for $u \to v \in \mathcal{P}_\mathcal{G}$) or $\theta(t)$ (for t such that $s \to t \in \mathcal{S}_\sharp$) using innermost *CSR*; as in the unrestricted case, the reachability problem for innermost *CSR* is undecidable. So, we need to use some approximation of it.

In [AGL10], we have adapted to the context-sensitive setting the more recent approximation for standard rewriting [GTS05]. Given a TRS $\mathcal{R}$ and a replacement map μ, we let $\mathrm{TCAP}^\mu_\mathcal{R}$ be as follows:

$$\begin{array}{rcl} \mathrm{TCAP}^\mu_\mathcal{R}(x) & = & y \text{ if } x \text{ is a variable, and} \\ \mathrm{TCAP}^\mu_\mathcal{R}(f(t_1,\ldots,t_k)) & = & \begin{cases} f([t_1]^f_1,\ldots,[t_k]^f_k) & \text{if } f([t_1]^f_1,\ldots,[t_k]^f_k) \text{ does not unify} \\ & \text{with } l \text{ for any } l \to r \text{ in } \mathcal{R} \\ y & \text{otherwise} \end{cases} \end{array}$$

where y is intended to be a new, fresh variable that has not yet been used and given a term s, $[s]^f_i = \mathrm{TCAP}^\mu_\mathcal{R}(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that l shares no variable with $f([t_1]^f_1,\ldots,[t_k]^f_k)$ when the unification is attempted. Function $\mathrm{TCAP}^\mu_\mathcal{R}$ is intended to provide a suitable approximation of the aforementioned $(\mathcal{R},\mu)$-reachability problems by means of *unification*.

Proposition 6 ([AGL10]) *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $\mu \in M_\mathcal{F}$. Let $t, u \in \mathcal{T}(\mathcal{F},\mathcal{X})$ be such that $\mathcal{V}ar(t)\cap\mathcal{V}ar(u) = \varnothing$. If $\theta(t) \hookrightarrow^* \theta(u)$ for some substitution θ, then $\mathrm{TCAP}^\mu_\mathcal{R}(t)$ and u unify.*

In contrast to standard (μ-)rewriting, in the innermost setting it is not necessary to rename multiple occurrences of variables since all variables are always instantiated to (μ-)normal forms and cannot be reduced. However, in innermost *CSR* we have to replace by fresh variables those ones that are μ-replacing in the

right hand side of the pair, v, but not in the left-hand side, u, since they are not μ-normalized. Moreover we need to substitute every subterm with a defined root symbol by fresh variables only if the term is not equal to a μ-replacing subterm of u or it unifies with the left-hand side of some rule in $\mathcal{R}$.

We define a new version of the function, $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}$, which is able to approximate the ICS graph by taking into account these particularities of innermost *CSR*.

Definition 14 *Given a TRS $\mathcal{R}$, a replacement map μ and a term u, we let $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}$ be as follows:*

$$\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}(x) = \begin{cases} y & \text{if } x \in \mathcal{X} \text{ and } x \notin \mathcal{V}ar^{\mu}(u) \\ x & \text{otherwise} \end{cases}$$

$$\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}(f(t_1,\ldots,t_k)) = \begin{cases} f([t_1]^f_1,\ldots,[t_k]^f_k) & \text{if } f([t_1]^f_1,\ldots,[t_k]^f_k) \text{ does not unify with } l \text{ for any} \\ & l \to r \text{ in } \mathcal{R} \text{ or it is equal to a } \mu\text{-replacing subterm of } u \\ y & \text{otherwise} \end{cases}$$

where y is intended to be a new, fresh variable that has not yet been used and given a term s, $[s]^f_i = \mathsf{iTCAP}^{\mu}_{\mathcal{R},u}(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that l shares no variable with $f([t_1]^f_1,\ldots,[t_k]^f_k)$ when the unification is attempted.

Since when connecting in a chain collapsing pairs we deal with rules in $\mathcal{S}_\sharp$ instead of pairs in $\mathcal{P}_\mathcal{G}$, we cannot look at the left hand side of the pairs. Therefore, for dealing with pairs in $\mathcal{P}_\mathcal{X}$, we have to approximate their arcs in the same way that for *CSR*since we do not store information about left-hand sides of the pairs from which the hidden terms are obtained. So, we have the following:

Definition 15 (Estimated Innermost Context-Sensitive Graph of Pairs) *Let $\mathcal{R} = (\mathcal{F}, R)$, $\mathcal{S} = (\mathcal{H}, S)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}\cup\mathcal{H}}$. The* estimated *ICS-graph associated to $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ (denoted $\mathsf{EIG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$) has $\mathcal{P}$ as the set of nodes and arcs which connect them as follows:*

1. *There is an arc from $u \to v \in \mathcal{P}_\mathcal{G}$ to $u' \to v' \in \mathcal{P}$ if $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}(v)$ and u' unify by some mgu σ such that $\sigma(u), \sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.*
2. *There is an arc from $u \to v \in \mathcal{P}_\mathcal{X}$ to $u' \to v' \in \mathcal{P}$ if there is $s \to t \in \mathcal{S}_\sharp$ such that $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(t)$ and u' unify by some mgu σ such that $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.*

Definition 16 (Correctness of the Estimated ICS-Graph of Pairs) *Let $\mathcal{R} = (\mathcal{F}, R)$, $\mathcal{S} = (\mathcal{H}, S)$ and $\mathcal{P} = (\mathcal{G}, P)$ be TRSs and $\mu \in M_{\mathcal{F}\cup\mathcal{G}\cup\mathcal{H}}$. The* estimated *ICS-graph associated to $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ (denoted $\mathsf{EIG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$) has $\mathcal{P}$ as the set of nodes and arcs which connect them as follows:*

1. *If there is an arc from $u \to v \in \mathcal{P}_\mathcal{G}$ to $u' \to v' \in \mathcal{P}$ and substitutions θ and θ' such that $\theta(v) \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \theta'(u')$, $\theta(u), \theta'(u') \in \mathsf{NF}_\mu(\mathcal{R})$ then $\mathsf{iTCAP}^{\mu}_{\mathcal{R},u}(v)$ and u' unify by some mgu σ such that $\sigma(u), \sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.*

2. *If there is an arc from* $u \to v \in \mathcal{P}_{\mathcal{X}}$ *to* $u' \to v' \in \mathcal{P}$ *and there is* $s \in \mathcal{NHT}(\mathcal{R},\mu)$ *such that* $\theta(s^\sharp) = \theta(t) = t'$ *and substitutions* θ *and* θ' *such that* $\theta(v) \xhookrightarrow{\Lambda}{}^*_{\mathcal{S}_{\rhd\mu},\mu} \circ \xhookrightarrow{\Lambda}_{\mathcal{S}_\sharp,\mu} t' \xhookrightarrow{!}_{\mathcal{R},\mu,\mathsf{i}} \theta'(u')$ *then there is* $s \to t \in \mathcal{S}_\sharp$ *such that* $\mathrm{TCAP}^\mu_{\mathcal{R}}(t)$ *and* u' *unify by some mgu* σ *such that* $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})$.

According to Definition 13, we would have the corresponding one for the *estimated* ICS-DG: $\mathsf{EIDG}(\mathcal{R},\mu) = \mathsf{EIG}(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$.

Example 9 *Consider the following TRS* $\mathcal{R}$ *[Zan97, Example 4]:*

$$\begin{aligned}
\mathtt{f}(x) &\to \mathtt{cons}(x,\mathtt{f}(\mathtt{g}(x)))\\
\mathtt{g}(\mathtt{0}) &\to \mathtt{s}(\mathtt{0})\\
\mathtt{g}(\mathtt{s}(x)) &\to \mathtt{s}(\mathtt{s}(\mathtt{g}(x)))\\
\mathtt{sel}(\mathtt{0},\mathtt{cons}(x,y)) &\to x\\
\mathtt{sel}(\mathtt{s}(x),\mathtt{cons}(y,z)) &\to \mathtt{sel}(x,z)
\end{aligned}$$

with $\mu(\mathtt{0}) = \varnothing$, $\mu(\mathtt{f}) = \mu(\mathtt{g}) = \mu(\mathtt{s}) = \mu(\mathtt{cons}) = \{1\}$, *and* $\mu(\mathtt{sel}) = \{1,2\}$. *Then,* $\mathsf{iDP}(\mathcal{R},\mu)$ *consists of the following pairs:*

$$\mathtt{G}(\mathtt{s}(x)) \to \mathtt{G}(x) \tag{7}$$
$$\mathtt{SEL}(\mathtt{s}(x),\mathtt{cons}(y,z)) \to \mathtt{SEL}(x,z) \tag{8}$$
$$\mathtt{SEL}(\mathtt{s}(x),\mathtt{cons}(y,z)) \to z \tag{9}$$

and the unhiding rules are: $\mathsf{unh}_{\rhd\mu}(\mathcal{R},\mu) = \{\mathtt{f}(x) \to x\}$ *and* $\mathsf{unh}_\sharp(\mathcal{R},\mu) = \{\mathtt{f}(\mathtt{g}(x)) \to \mathtt{F}(g(x)), \mathtt{g}(x) \to \mathtt{G}(x)\}$.

Regarding pairs (7) *and* (8) *in* $\mathsf{iDP}_{\mathcal{F}}(\mathcal{R},\mu)$, *there is an arc from* (7) *to itself and another one from* (8) *to itself. Regarding the only collapsing pair* (9), *we have* $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{F}(\mathtt{g}(x))) = \mathtt{F}(y)$ *and* $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{G}(x)) = \mathtt{G}(y)$. *Since* $\mathtt{F}(y)$ *does not unify with the left-hand side of any pair, and* $\mathtt{G}(y)$ *unifies with the left-hand side* $\mathtt{G}(\mathtt{s}(x))$ *of* (7) *and* $\mathtt{G}(\mathtt{s}(x))$ *is in* μ*-normal form, there is an arc from* (9) *to* (7), *see Figure 1. Thus, there are two cycles:* $\{(7)\}$ *and* $\{(8)\}$.

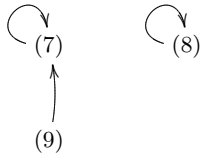


Figure 1: Innermost CS-Dependency graph for Example 9

The following example shows that using $\mathrm{iTCAP}^\mu_{\mathcal{R},u}$ provides a better approximation of the ICS-DG than using $\mathrm{TCAP}^\mu_{\mathcal{R}}$ for noncollapsing pairs.

Example 10 *Consider the following TRS* $\mathcal{R}$*:*

$$\begin{array}{rcl} \mathtt{f}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{f}(x,x,x) \\ \mathtt{c} & \to & \mathtt{a} \\ \mathtt{c} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$. *There are two ICS-dependency pairs:*

$$\begin{array}{rcl} \mathtt{F}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{F}(x,x,x) \\ \mathtt{F}(\mathtt{a},\mathtt{b},x) & \to & x \end{array}$$

$\mathcal{R}$ *is not innermost* μ*-terminating:*

$$\mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \mathtt{F}(\mathtt{a},\underline{\mathtt{c}},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \underline{\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c})} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu,\mathsf{i}),\mu^\sharp} \mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \cdots$$

In order to build the ICS-DG, since there are not hidden terms and therefore $\mathcal{S}_\sharp$ *is empty, we only have to check if* $\mathsf{iTCAP}^\mu_{\mathcal{R},u}(\mathtt{F}(x,x,x)) = \mathsf{iTCAP}^\mu_{\mathcal{R},\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{x})}(\mathtt{F}(x,x,x)) = \mathtt{F}(x''',x'',x)$ *unifies with* $\mathtt{F}(\mathtt{a},\mathtt{b},y)$ *so, we get a cycle and the same would be obtained with* $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{F}(x,x,x))$. *However, if we use* $\mu(\mathtt{f}) = \{1,3\}$, *the system now is innermost* μ*-terminating (the collapsing pair now disappears) but if we use the* $\mathrm{TCAP}^\mu_{\mathcal{R}}$ *we get* $\mathrm{TCAP}^\mu_{\mathcal{R}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x''',x'',x)$ *again unifies with* $\mathtt{F}(\mathtt{a},\mathtt{b},x)$ *and we obtain a spurious cycle. By using* $\mathsf{iTCAP}^\mu_{\mathcal{R},u}$, *we obtain* $\mathsf{iTCAP}^\mu_{\mathcal{R},\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{x})}(\mathtt{F}(x,x,x)) = \mathtt{F}(x,x,x)$ *(since there are not migrating variables now) which does not unify with* $\mathtt{F}(\mathtt{a},\mathtt{b},y)$. *Now, innermost* μ*-termination can be easily proved since there are no cycles in the ICS-DG.*

After showing that $\mathsf{iTCAP}^\mu_{\mathcal{R},u}$ provides a better approximation of the ICS-DG for noncollapsing pairs, we are going to show that for the collapsing pairs this is not true since we can lead into and underestimation of the graph and conclude a false result.

Example 11 *Consider the following TRS* $\mathcal{R}$ *which is a variant of Example 10:*

$$\begin{array}{rcl} \mathtt{f}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{g}(\mathtt{f}(x,x,x)) \\ \mathtt{g}(x) & \to & x \\ \mathtt{c} & \to & \mathtt{a} \\ \mathtt{c} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1,2\}$ *and* $\mu(\mathtt{g}) = \varnothing$. *There are two ICS-dependency pairs:*

$$\begin{array}{rcll} \mathtt{F}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{G}(\mathtt{f}(x,x,x)) & (10) \\ \mathtt{G}(x) & \to & x & (11) \end{array}$$

$\mathcal{R}$ *is not innermost* μ*-terminating:*

$$\mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \mathtt{F}(\mathtt{a},\underline{\mathtt{c}},\mathtt{c}) \hookrightarrow_{\mathcal{R},\mu^\sharp,\mathsf{i}} \underline{\mathtt{F}(\mathtt{a},\mathtt{b},\mathtt{c})} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu),\mu^\sharp,\mathsf{i}} \underline{\mathtt{G}(\mathtt{f}(\mathtt{a},\mathtt{b},\mathtt{c}))} \hookrightarrow_{\mathsf{iDP}(\mathcal{R},\mu),\mu^\sharp,\mathsf{i}} \mathtt{F}(\underline{\mathtt{c}},\mathtt{c},\mathtt{c}) \cdots$$

We have $\mathcal{S}_\sharp = \{\mathtt{f}(x,x,x) \to \mathtt{F}(x,x,x)\}$. *Regarding the pair* (10) $\in \mathsf{iDP}_{\mathcal{F}}(\mathcal{R},\mu)$, *there is an obvious arc from* (10) *to* (11). *Regarding the only collapsing pair* (11), *since we do not have any information in* $\mathcal{S}_\sharp$ *about migrating variables, we have*

to use $\mathrm{TCAP}^{\mu}_{\mathcal{R}}$. *In this way, we have that* $\mathrm{TCAP}^{\mu}_{\mathcal{R}}(\mathtt{F}(x,x,x)) = \mathtt{F}(x'',x',x)$ *unifies with* $\mathtt{F}(\mathtt{a},\mathtt{b},y)$ *and we obtain an arc from* (11) *to* (10)*, thus obtaining the existing cycle* {(11)-(10)}. *Otherwise, we will not rename any variable and we would not have obtained the arc.*

Example 12 *(Continuing Example 7) Since* $\mathrm{iTCAP}^{\mu}_{\mathcal{R},\mathtt{F}(c(x),x)}(\mathtt{F}(x,x)) = \mathtt{F}(x,x)$ *and* $\mathtt{F}(c(y),y)$ *do not unify we conclude (and this can easily be implemented) that the ICS-dependency graph for the CS-TRS* $(\mathcal{R},\mu)$ *in Example 7 contains no cycles.*

Since for approximating the innermost context-sensitive graph of a set of pairs, we use function $\mathrm{TCAP}^{\mu}_{\mathcal{R}}$ for connecting pairs in $\mathcal{P}_{\mathcal{X}}$ as done in the context-senstive case, we can also use the following processor instead, which allows a better approximation of the SCCs. This is because if the SCC has no collapsing pairs, the set $\mathcal{S}$ has no sense and if it has, some pairs from $\mathcal{S}_\sharp$ can be removed: those that are not involved in the unification process. Therefore, we will always compute the SCCs by applying the following processor:

Theorem 10 (SCC Processor using $\mathrm{TCAP}^{\mu}_{\mathcal{R}}$ **[GL10])** *Let* $\tau = (\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$ *be a CS problem. The CS processor* Proc_{SCC} *given by*

$$\mathsf{Proc}_{SCC}(\tau) = \{(\mathcal{Q},\mathcal{R},\mathcal{S}_{\mathcal{Q}},\mu) \mid \mathcal{Q} \text{ contains the pairs of an SCC in } \mathsf{EIG}(\mathcal{P},\mathcal{R},\mathcal{S},\mu)\}$$

where

- $\mathcal{S}_{\mathcal{Q}} = \varnothing$ *if* $\mathcal{Q}_{\mathcal{X}} = \varnothing$.
- $\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\rhd_\mu} \cup \{s \to t \mid s \to t \in \mathcal{S}_\sharp, \mathrm{TCAP}^{\mu}_{\mathcal{R}}(t)$ *and* u' *unify for some* $u' \to v' \in \mathcal{Q}$ *by some mgu* σ *such that* $\sigma(u') \in \mathsf{NF}_\mu(\mathcal{R})\}$ *if* $\mathcal{Q}_{\mathcal{X}} \neq \varnothing$.

is sound and complete.

Example 13 *Consider again Example 1. The set* $\mathsf{iDP}(\mathcal{R},\mu)$ *is in Example 5 and the unhiding TRS* $\mathsf{unh}(\mathcal{R},\mu)$ *consists of the rules in Example 6. We can define the following CS problem:*

$$\tau_0 = (\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp,\mathbf{i})$$

The $\mathsf{EIDG}(\mathcal{R},\mu) = \mathsf{EIG}(\mathsf{iDP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$ *of the CS problem* τ_0 *is shown in Figure 2. If now we apply the improved SCC processor we get the followings CS subproblems:*

$$\mathsf{Proc}_{SCC}(\tau_0) = \{(\{1\},\mathcal{R},\varnothing,\mu^\sharp,\mathbf{i}),(\{3\},\mathcal{R},\varnothing,\mu^\sharp,\mathbf{i}),(\{4\},\mathcal{R},\varnothing,\mu^\sharp,\mathbf{i})\}$$

8 Usable Rules

An interesting feature in the treatment of innermost termination problems using the dependency pair approach is that, since the variables in the right-hand side of the dependency pairs are in normal form, the rules which can be used to

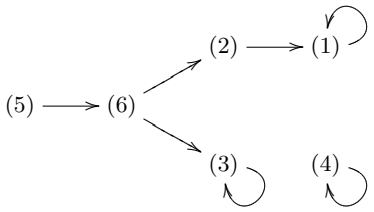


Figure 2: Estimated Innermost CS-Dependency Graph for Example 1

connect contiguous dependency pairs are usually a proper subset of the rules in the TRS. This leads to the notion of *usable rules* [AG00, Definition 32] which simplifies the proofs of innermost termination of rewriting. We adapt this notion to the context-sensitive setting.

Definition 17 (Basic usable CS-rules) *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For any symbol f let $Rules(\mathcal{R}, f)$ be the set of rules of $\mathcal{R}$ defining f and such that the left-hand side l has no proper μ-replacing $\mathcal{R}$-redex. For any term t, the set of* basic *usable rules $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is as follows:*

$$\begin{array}{lcl} \mathbf{U}_0(\mathcal{R}, \mu, x) & = & \varnothing \\ \mathbf{U}_0(\mathcal{R}, \mu, f(t_1, \ldots, t_n)) & = & Rules(\mathcal{R}, f) \cup \bigcup_{i \in \mu(f)} \mathbf{U}_0(\mathcal{R}', \mu, t_i) \cup \bigcup_{l \to r \in Rules(\mathcal{R}, f)} \mathbf{U}_0(\mathcal{R}', \mu, r) \end{array}$$

where $\mathcal{R}' = \mathcal{R} - Rules(\mathcal{R}, f)$.

Consider now a TRS $\mathcal{P}$. Then, $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) = \bigcup_{l \to r \in \mathcal{P}} \mathbf{U}_0(\mathcal{R}, \mu, r)$. Obviously, $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) \subseteq \mathcal{R}$ for all TRSs $\mathcal{P}$ and $\mathcal{R}$.

Interestingly, although our definition is a straightforward extension of the classical one (which just takes into account that μ-rewritings are possible only on μ-replacing subterms), some subtleties arise due to the presence of *non-conservative* rules.

Basic usable rules $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ in Definition 17 can be used instead of $\mathcal{R}$ when dealing with innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathsf{i})$-chains associated to μ-conservative TRSs $\mathcal{P}$ provided that $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ is also μ-conservative. This is proved in Theorem 11 below. First, we need some auxiliary results.

Proposition 7 *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $s \hookrightarrow_{\mathsf{i}} s'$ by applying a rule $l \to r \in \mathcal{R}$, then there is a substitution σ' such that $s' = \sigma'(t')$ for $t' = t[r]_p$ and $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$.*

Proof. Let $p \in \mathcal{P}os^{\mu}(s)$ be the position of an innermost redex $s|_p = \theta(l)$ for some substitution θ. Since $s = \sigma(t)$ and for all replacing variables in t, we have $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$, it follows that p is a non-variable (replacing) position of t. Therefore, $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Since $s = \sigma(t)$, we have that $s' = \sigma(t)[\theta(r)]_p$ and since

$p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$, by defining $\sigma'(x) = \sigma(x)$ for all $x \in Var(t)$ and $\sigma(x) = \theta(x)$ for all $x \in \mathcal{V}ar(r)$ (as usual, we assume $\mathcal{V}ar(t) \cap \mathcal{V}ar(r) = \varnothing$), we have $s' = \sigma'(t[r]_p)$. □

The following proposition states that an innermost μ-rewrite step by applying a conservative rule makes the set of μ-replacing variables of the contractum will be instantiated to μ-normal forms.

Proposition 8 *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $s \hookrightarrow_{\mathrm{i}} s'$ by applying a conservative rule $l \to r \in \mathcal{R}$, then there is a substitution σ' such that $s' = \sigma'(t')$ for $t' = t[r]_p$, $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$.*

Proof. By Proposition 7, we know that σ', as in Proposition 7, satisfies $s' = \sigma'(t')$ for θ as in Proposition 7 and some $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Since $s|_p$ is an innermost μ-replacing redex, we have that $\forall y \in \mathcal{V}ar^{\mu}(l)$, $\theta(y) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since the rule $l \to r$ is conservative, $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$, hence $\forall z \in \mathcal{V}ar^{\mu}(r)$, $\sigma'(z) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since $\mathcal{V}ar^{\mu}(t[r]_p) \subseteq \mathcal{V}ar^{\mu}(t) \cup \mathcal{V}ar^{\mu}(r)$, we have that $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. □

Now, we prove that in an innermost μ-rewrite sequence starting from a term instantiated with a μ-normalized substitution, the only rules that can be applied are the usable rules (if they are μ-conservative).

Proposition 9 *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. Let $t, s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and σ be a substitution such that $s = \sigma(t)$ and $\forall x \in \mathcal{V}ar^{\mu}(t)$, $\sigma(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. If $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is conservative and $s = s_1 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} s_2 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} \cdots \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} s_n \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} s_{n+1} = u$ for some $n \geq 0$ then $s_i \hookrightarrow_{\mathbf{U}_0(\mathcal{R},\mu,t),\mu,\mathrm{i}} s_{i+1}$ for all i, $1 \leq i \leq n$.*

Proof. By induction on n. If $n = 0$, then $s = \sigma(t) = u$, it is trivial. Otherwise, if $s_1 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} s_2 \hookrightarrow^{*}_{\mathcal{R},\mu,\mathrm{i}} u$, we first prove that the result also holds in $s_1 \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} s_2$. By Proposition 7, $s_1 = \sigma(t)$, and $s_2 = \sigma'(t')$ for $t' = t[r]_p$ is such that $s_1|_p = \theta(l)$ and $s_2|_p = \theta(r)$ for some $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(t)$. Thus, $root(l) = root(t|_p)$ and by Definition 17, we can conclude that $l \to r \in \mathbf{U}_0(\mathcal{R}, \mu, t)$. By hypothesis, $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is conservative. Thus, $l \to r$ is conservative and by Proposition 8, $s_2 = \sigma'(t')$ and $\forall x \in \mathcal{V}ar^{\mu}(t')$, $\sigma'(x) \in \mathsf{NF}_{\mu}(\mathcal{R})$. Since $t' = t[r]_p$ and $root(t|_p) = root(l)$, we have that $\mathbf{U}_0(\mathcal{R}, \mu, t') \subseteq \mathbf{U}_0(\mathcal{R}, \mu, t)$ and (since $\mathbf{U}_0(\mathcal{R}, \mu, t)$ is conservative) $\mathbf{U}_0(\mathcal{R}, \mu, t')$ is conservative as well. By the induction hypothesis we know that $s_i \hookrightarrow_{\mathbf{U}_0(\mathcal{R},\mu,t'),\mu,\mathrm{i}} s_{i+1}$ for all i, $2 \leq i \leq n$. Thus we have $s_i \hookrightarrow_{\mathbf{U}_0(\mathcal{R},\mu,t),\mu,\mathrm{i}} s_{i+1}$ for all i, $1 \leq i \leq n$ as desired.

□

The following theorem formalizes a processor to remove pairs from $\mathcal{P}$ by using the previous result and μ-reduction pairs.

Theorem 11 *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $(\gtrsim, \sqsupset)$ be a μ-reduction pair such that*

1. $\mathcal{P}$ *and* $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$ *are conservative,*

2. $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}) \subseteq \gtrsim$ *and* $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$,

Let $\mathcal{P}_\sqsupset = \{u \to v \in \mathcal{P} \mid u \sqsupset v\}$. *Then, the processor* Proc_{UR} *given by*

$$\mathsf{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}), \varnothing, \mu, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound.

Proof.

We proceed by contradiction. Assume that there is an infinite minimal innermost $(\mathcal{P},\mathcal{R},\mathcal{S},\mu,\mathbf{i})$-chain A, but that there is no infinite minimal innermost $(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}), \varnothing, \mu, \mathbf{i})$-chain. Due to the finiteness of $\mathcal{P}$, and since $\mathcal{P}$ is conservative, we have $\mathcal{P}_\mathcal{X} = \varnothing$. Thus, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B

$$\sigma(u_1) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} t_1 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_2) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} t_2 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_3) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} \cdots$$

for some substitution σ, where all pairs in $\mathcal{Q}$ are infinitely often used, and, for all $i \geq 1$, since all $u_i \to v_i \in \mathcal{P}$ are conservative $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$ (i.e. $\mathcal{P}_\mathcal{X} = \mathcal{Q}_\mathcal{X} = \varnothing$), then $t_i = \sigma(v_i)$ and $\sigma(u_i) \in \mathsf{NF}_\mu(\mathcal{R})$, this implies that $\forall x \in \mathcal{V}ar^\mu(u_i)$, $\sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$ and by Proposition 9 the sequence can be seen as:

$$\sigma(u_1) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} t_1 \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu,\mathsf{i}} \sigma(u_2) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} t_2 \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu,\mathsf{i}} \sigma(u_3) \hookrightarrow_{\mathcal{Q}_\mathcal{G},\mu} \cdots$$

Furthermore, by minimality, $t_i = \sigma(v_i)$ is innermost $(\mathcal{R},\mu)$-terminating for all $i \geq 1$. Since $u_i \,(\gtrsim \cup \sqsupset)\, v_i$ for all $u_i \to v_i \in \mathcal{Q} \subseteq \mathcal{P}$, by stability of $\gtrsim$ and $\sqsupset$, we have $\sigma(u_i)\,(\gtrsim \cup \sqsupset)\,\sigma(v_i)$ for all $i \geq 1$. No pair $u \to v \in \mathcal{Q}$ satisfies that $u \sqsupset v$. Otherwise, we get a contradiction by considering that since all pairs in $\mathcal{P}$ are conservative, we have that $u_i \to v_i \in \mathcal{Q}_\mathcal{G}$. Then, $t_i = \sigma(v_i) \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu,\mathsf{i}} \sigma(u_{i+1})$ and $t_i \gtrsim \sigma(u_{i+1})$. Since we have $\sigma(u_i)\,(\gtrsim \cup \sqsupset)\,\sigma(v_i)$, by using transitivity of $\gtrsim$ and compatibility between $\gtrsim$ and $\sqsupset$, we conclude that $\sigma(u_i)\,(\gtrsim \cup \sqsupset)\,\sigma(u_{i+1})$. Since $u \to v$ occurs infinitely often in B, there is an infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that $\sigma(u_i) \sqsupset \sigma(u_{i+1})$ for all $i \in \mathcal{I}$. And we have $\sigma(u_i)\,(\gtrsim \cup \sqsupset)\,\sigma(u_{i+1})$ for all other $u_i \to v_i \in \mathcal{Q}$. Thus, by using the compatibility conditions of the μ-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence which contradicts well-foundedness of $\sqsupset$. Therefore, $\mathcal{Q} \subseteq (\mathcal{P} \setminus \mathcal{P}_\sqsupset)$. Since $\mathsf{NF}_\mu(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})) \supseteq \mathsf{NF}_\mu(\mathcal{R})$, we have that $\sigma(u_i) \in \mathsf{NF}_\mu(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}))$. By Proposition 9, innermost $(\mathcal{R},\mu)$-termination of $\sigma(v_i)$ implies innermost $(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu)$-termination of $\sigma(v_i)$ for all $i \geq 1$. Hence, B is an infinite minimal innermost $(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}), \varnothing, \mu, \mathbf{i})$-chain, thus leading to a contradiction. □

Note that the processor is only sound because we refine the result to be applied only to the set of usable rules instead of over the whole set of rules as

in standard rewriting [GTSF06] or even for context-sensitive in [AEF+08]. In this way, (i.e. by taking all the rules in $\mathcal{R}$), it would be also complete, that is:

$$\mathsf{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{R}, \varnothing, \mu, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound and complete, but since complete processors are useful for disproving termination, we pay more attention on be more precise with the soundness.

Note also, that in the case of usable rules for context-sensitive rewriting (non innermost) [GLU08], this improvement is not possible to be taken into consideration, since it would be unsound.

Unfortunately, dealing with nonconservative pairs, considering the basic usable CS-rules does *not* ensure a correct approach.

Example 14 *Consider again the TRS $\mathcal{R}$:*

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c(b)} \\ \mathtt{f(c}(x)\mathtt{,}x\mathtt{)} & \to & \mathtt{f}(x,x) \end{array}$$

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{c}) = \varnothing$. There are two *non-conservative ICS-DPs (note that $\mu^{\sharp}(\mathtt{F}) = \mu(\mathtt{f}) = \{1\}$):*

$$\begin{array}{rcl} \mathtt{F(c}(x)\mathtt{,}x\mathtt{)} & \to & \mathtt{F}(x,x) \\ \mathtt{F(c}(x)\mathtt{,}x\mathtt{)} & \to & x \end{array}$$

and only one cycle in the ICS-DG:

$$\{\mathtt{F(c}(x)\mathtt{,}x\mathtt{)} \to \mathtt{F}(x,x)\}$$

Note that $\mathbf{U}_0(\mathcal{R}, \mu, \mathtt{F}(x,x)) = \varnothing$. Since this ICSDP is strictly compatible with, e.g., an LPO, we would conclude the innermost μ-termination of $\mathcal{R}$. However, this system is not *innermost μ-terminating:*

$$\mathtt{f}(\underline{\mathtt{b}}, \mathtt{b}) \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f(c(b)}, \mathtt{b})} \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{b}}, \mathtt{b}) \hookrightarrow_{\mathsf{i}} \cdots$$

The problem is that we have to take into account the special status of variables in the right-hand side of a nonconservative ICSDP. Instances of such variables are *not* guaranteed to be μ-normal forms. Furthermore, conservativeness of $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ cannot be dropped either since we could infer an incorrect result as shown by the following example.

Example 15 *Consider the TRS $\mathcal{R}$:*

$$\begin{array}{rcl} \mathtt{b} & \to & \mathtt{c(b)} \\ \mathtt{f(c}(x)\mathtt{,}x\mathtt{)} & \to & \mathtt{f(g}(x)\mathtt{,}x\mathtt{)} \\ \mathtt{g}(x) & \to & x \end{array}$$

together with $\mu(\mathtt{f}) = \{1\}$ and $\mu(\mathtt{g}) = \mu(\mathtt{c}) = \varnothing$. There is only one conservative cycle:

$$\{\mathtt{F(c}(x)\mathtt{,}x\mathtt{)} \to \mathtt{F(g}(x)\mathtt{,}x\mathtt{)}\}$$

having only one usable (but non-conservative!) rule $\mathtt{g}(x) \to x$ *This is compatible with the* μ*-reduction pair induced by the following polynomial interpretation:*

$$[\mathtt{f}](x,y) = 0 \qquad [\mathtt{c}](x) = x+1 \qquad [\mathtt{g}](x) = x \qquad [\mathtt{F}](x,y) = x$$

However the system is not innermost μ*-terminating:*

$$\underline{\mathtt{f(c(b),b)}} \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{g(b)}},\mathtt{b}) \hookrightarrow_{\mathsf{i}} \mathtt{f}(\underline{\mathtt{b}},\mathtt{b}) \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f(c(b),b)}} \hookrightarrow_{\mathsf{i}} \cdots$$

Nevertheless, Theorem 11 is useful to improve the proofs of termination of innermost *CSR* as the following example shows.

Example 16 *Consider again the TRS* $\mathcal{R}$ *in Example 1. As we have seen in Example 13 the initial CS problem can be decomposed in the following three:*

$$\tau_1 = (\{1\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{12}$$
$$\tau_2 = (\{3\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{13}$$
$$\tau_3 = (\{4\}, \mathcal{R}, \varnothing, \mu^\sharp, \mathbf{i}) \tag{14}$$

Problems τ_1 *and* τ_3 *can be solved by using the subterm processor (see [GL10]). However, without the notion of usable rules,* τ_2 *is difficult to solve. The pair (3) is* μ*-conservative and the obtained usable rules are also* μ*-conservative:*

$$\mathtt{minus}(x,0) \to x$$

and

$$\mathtt{minus}(\mathtt{s}(x),\mathtt{s}(x)) \to \mathtt{minus}(x,y)$$

According to Theorem 11, we can apply the usable rules processor $\mathsf{Proc}_{UR}(\tau_2)$ *and get the following problem:*

$$\tau_4 = (\varnothing, \{\mathtt{minus}(x,0) \to x, \mathtt{minus}(\mathtt{s}(x),\mathtt{s}(x)) \to \mathtt{minus}(x,y)\}, \varnothing, \mu^\sharp, \mathbf{i})$$

by using a polynomial interpretation:

$$[\mathtt{minus}](x,y) = x \qquad [\mathtt{0}] = 0$$
$$[\mathtt{s}](x) = x+1 \qquad [\mathtt{QUOT}](x,y) = x$$

Then, by applying $\mathsf{Proc}_{Fin}(\tau_4)$*, since the set of pairs is empty, we can conclude the innermost* μ*-termination of Example 1. Furthermore, since Example 1 is orthogonal, we have also concluded its* μ*-termination.*

9 Usable Arguments for *CSR*

Since in innermost reductions, matching substitutions are always normalized, in an innermost sequence $t_1 \xrightarrow{p_1}_i t_2 \xrightarrow{p_2}_i \cdots \xrightarrow{p_n}_i t_{n+1}$ starting at root position (i.e., $p_1 = \Lambda$), every redex $t_j|_{p_j}$ for $j > 1$ comes from a defined symbol introduced after applying a rule $l_k \to r_k$ in a previous step $k < j$. Hence the set of arguments which are reduced can be handled by looking for defined symbols in right-hand sides of the involved rules $l \to r$.

In [Fer05] Fernández defines the notion of *usable arguments* for a function symbol when proving innermost termination. The idea is that, in innermost sequences, some arguments are not relevant for proving termination.

Example 17 *Consider the following TRS $\mathcal{R}$:*

$$\begin{array}{rcl} \mathtt{f}(\mathtt{s}(x), \mathtt{s}(x)) & \to & \mathtt{f}(x, \mathtt{g}(x)) \\ \mathtt{g}(\mathtt{s}(x)) & \to & \mathtt{g}(x) \end{array}$$

No innermost sequence starting at root position takes into account the first argument of $\mathtt{f}$ *nor the argument of* $\mathtt{g}$*. The reason is that since an innermost redex is an argument normalized redex, that means that all variables (e.g. x) of the applied rule are normalized and cannot be reduced. Only the second argument* $\mathtt{g}(x)$ *of* $\mathtt{f}$ *in the right-hand side of the first rule could be innermost reduced after applying it.*

Considering those usable arguments could be helpful in proofs of innermost termination since they impose weaker monotonicity requirements. For instance, when using polynomial orderings, we can use even negative or rational coefficients for interpreting the symbols that do not need to be monotonic.

As Fernández noticed in [Fer05], the set of usable arguments can be seen as a replacement map which specifies the arguments to be reduced. In her approach, proving the μ-termination of a TRS $\mathcal{R}$ implies the innermost termination of $\mathcal{R}$ if $\mu(f)$=$\mathbf{UA}(f, \mathcal{R}, R)$ for all $f \in \mathcal{F}$ where R only contains rules such that all left-hand sides are argument normalized.

Following Fernández's ideas, in the innermost context-sensitive setting (for a given replacement map μ) we could relax monotonicity requirements by taking into account that reductions only take place on μ-replacing positions of the right-hand sides of the rules which are rooted by a defined symbol.

We have adapted Fernández's ideas to *CSR* in [AL09]. In sharp contrast to the unrestricted case, we need to take into account that in innermost *CSR* a redex does not need to be argument normalized. Only argument μ-normalization can be assumed. Thus, non-μ-replacing subterms may contain redexes that can be reduced later on if they come to a replacing position.

Proposition 10 *A CS-TRS $(\mathcal{R}, \mu)$ is innermost μ-terminating iff $\mathcal{R}'$ is innermost μ-terminating, where $\mathcal{R}' \subseteq \mathcal{R}$ contains all rules $l \to r \in \mathcal{R}$ such that l is argument μ-normalized.*

PROOF. Trivial since the only rules that can be applied in innermost μ-reductions are those whose the left-hand sides are argument μ-normalized as we have shown in the Definition 5 of ICSDPs. □

In the following, we assume that all CS-TRS $(\mathcal{R}, \mu)$ are argument μ-normalized, i.e., for all rule $l \to r$ in $\mathcal{R}$, l is argument μ-normalized. Proposition 10 ensures that this entails no lack of generality regarding our research on innermost termination of *CSR*.

The straightforward adaptation of Fernandez's criterion to *CSR* yields the following definition: the usable CS-arguments for a function symbol $f \in \mathcal{F}$ are those arguments with a μ-replacing subterm rooted by a defined symbol in some right-hand side of a pair or usable rule.

Definition 18 (Basic usable CS-arguments) *Let $(\mathcal{R}, \mu) = ((\mathcal{C} \uplus \mathcal{D}, R), \mu)$ be a CS-TRS and $\mathcal{P}$ be a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument μ-normalized. The* basic usable CS-arguments *for a function symbol $f \in \mathcal{F}$ are defined as $\mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P}) = \{i \in \mu(f) \mid \exists u \to v \in \mathcal{P} \cup \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}),$ $\exists p, p' \in \mathcal{P}os^\mu(v)$ s.t. $root(v|_{p'}) = f$, $root(v|_p) \in \mathcal{D}$, $p'.i \leqslant p$, $u \not\trianglerighteq_\mu v|_p\}$.*

Note that the replacement map given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for all $f \in \mathcal{F}$ is more restrictive than μ: $\mu'(f) \subseteq \mu(f)$ for all $f \in \mathcal{F}$.

The following proposition is the context-sensitive version of [Fer05, Lemma 5].

Proposition 11 *Let $(\mathcal{R}, \mu)$ be a CS-TRS and $\mathcal{P}$ be a set of pairs of terms s.t. for all $u \to v \in \mathcal{P}$, u is argument μ-normalized and $\mathcal{P} \cup \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ is μ-conservative. Let $l \to r \in \mathcal{P} \cup \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ be such that $\sigma(r) \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}\,\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})} t$ for some term t and substitution σ s.t. $\sigma(l)$ is argument μ-normalized . If $t|_p$ is an innermost μ-redex, then for all $p'.k \leqslant p$, we have that $k \in \mathbf{UA}_\mu(root(t|_{p'}), \mathcal{R}, \mathcal{P})$.*

PROOF. By induction on the length n of the rewriting sequence. If $n = 0$, then $\sigma(r) = t$. Then, since $\sigma(l)$ is argument μ-normalized, it follows that for all $x \in \mathcal{V}ar^\mu(l), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$. Since the rule $l \to r$ is conservative (that is $\mathcal{V}ar^\mu(r) \subseteq \mathcal{V}ar^\mu(l)$), we have that for all $x \in \mathcal{V}ar^\mu(r), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$. It follows that p is a nonvariable (μ-replacing) position of r, i.e. $p \in \mathcal{P}os^\mu_{\mathcal{F}}(r)$. Thus, $\mathrm{root}(r|_p) \in \mathcal{D}$ and the result follows by Definition 18.

If $n > 0$, then there is a term s such that $\sigma(r) \overset{>\Lambda}{\hookrightarrow}^*_{\mathsf{i}} s$ and $s \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} t$ at some μ-replacing position q. By the induction hypothesis, every μ-replacing position of the term t above, which equal or disjoint to q satisfies the result and we only have to prove it for innermost redexes $t|_p$ s.t. $q < p$, it is say, we have to prove that $k \in \mathbf{UA}_\mu(root(t|_{p'}), \mathcal{R}, \mathcal{P})$, for all $q < p'.k \leqslant p$. If $s \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} t$, then $s|_q = \sigma'(l')$ and $t|_q = \sigma'(r')$, for some rule $l' \to r' \in \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ and substitution σ' s.t. $\sigma'(l')$ is argument μ-normalized. This implies that every innermost redex of $t|_q$ occurs at a position $p'' \in \mathcal{P}os^\mu(r')$ s.t. $root(r'|_{p''}) \in \mathcal{D}$ (since the rule $l' \to r'$ is conservative we have that for all $x \in \mathcal{V}ar^\mu(r'), \sigma(x) \in \mathsf{NF}_\mu(\mathcal{R})$) and $l' \not\trianglerighteq_\mu r'|_{p''}$(otherwise, $\sigma'(l')$ would not be an innermost redex of

s. By definition, when $p'' > \Lambda$, $p'.k \leqslant p''$, $k \in \mathbf{UA}_\mu(root(t|_{q.p'}), \mathcal{R}, \mathcal{P})$ which is equivalent to what we needed to prove ($k \in \mathbf{UA}_\mu(root(t|_{p'}), \mathcal{R}, \mathcal{P})$, for all $q < p'.k \leqslant p$). □

Corollary 11 in [Fer05] suggests that innermost μ-termination could be proved by using a μ'-reduction ordering for μ' given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for all $f \in \mathcal{F}$. This is true for μ'-conservative CS-TRSs, as the following theorem shows.

Theorem 12 *A μ-conservative CS-TRS $(\mathcal{R}, \mu)$ is innermost μ-terminating if there is a μ'-reduction ordering $\succ$ s.t. $\mathcal{R} \subseteq \succ$, where for all symbol $f \in \mathcal{F}$, $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, R)$.*

Proof. By contradiction. Assume that $\mathcal{R}$ is not innermost μ-terminating. By the argument of size minimality, there is a infinite innermost μ-rewrite sequence with the first step at position Λ: $s_1 \hookrightarrow_{\mathsf{i}} s_2 \hookrightarrow_{\mathsf{i}} s_3 \hookrightarrow_{\mathsf{i}} \cdots$ (without loss of generality). By Proposition 11 (where we let $\mathcal{P} = \mathcal{R}$), every step $s_j \overset{>\Lambda}{\hookrightarrow}_{\mathsf{i}} s_{j+1}$ at position p satisfies that $p'.k \leqslant p$, $k \in \mathbf{UA}_\mu(root(s_j|_{p'}), \mathcal{R}, \mathcal{P})$. Since $\mathcal{R} \subseteq \succ$ and $\succ$ is stable and μ'-monotonic, $s_j \succ s_{j+1}$ holds. Therefore, there is an infinite $\succ$-decreasing sequence of terms $s_1 \succ s_2 \succ \cdots \succ s_n \succ \cdots$ which contradicts the well-foundedness of $\succ$.

□

Since μ-reduction orderings characterize termination of *CSR* we have the following corollary.

Corollary 4 *Let $\mathcal{R}$ be a μ-conservative TRS for $\mu \in M_\mathcal{R}$. Let μ' be given by $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, R)$ for every $f \in \mathcal{F}$. If $\mathcal{R}$ is innermost μ'-terminating, then $\mathcal{R}$ is innermost μ-terminating.*

Example 18 *Consider the TRS $\mathcal{R}$:*

$$\begin{array}{rcl} \mathtt{f}(\mathtt{a},\mathtt{b},x) & \to & \mathtt{f}(x,x,x) \\ \mathtt{c} & \to & \mathtt{a} \\ \mathtt{c} & \to & \mathtt{b} \end{array}$$

together with $\mu(\mathtt{f}) = \{1, 3\}$. By using $\mu'(f) = \mathbf{UA}_\mu(f, \mathcal{R}, \mathcal{P})$ for every $f \in \mathcal{F}$ we obtain $\mu'(\mathtt{f}) = \varnothing$. The pair $\mathtt{f}(\mathtt{a}, \mathtt{b}, x) \to \mathtt{f}(x, x, x)$ cannot form a cycle now, thus easily concluding the μ'-termination of $\mathcal{R}$ and, by Corollary 4, the innermost μ-termination of $\mathcal{R}$.

This fact is important since now, all techniques for proving termination of *CSR* can be used to prove termination of innermost *CSR* for μ-conservative systems. The following example shows that μ-conservativeness cannot be dropped in Theorem 12 and Corollary 4.

Example 19 *Consider again the TRS $\mathcal{R}$ in Example 18 but now together with $\mu(\mathtt{f}) = \{1, 2\}$. If we try to apply Corollary 4 to prove innermost μ-termination*

of $\mathcal{R}$, we obtain $\mu'(\mathtt{f}) = \varnothing$ and (as discussed in Example 18) the CS-dependency graph has no cycle thus concluding the innermost μ-termination of $\mathcal{R}$. However,$\mathcal{R}$ is not *innermost μ-terminating:*

$$\underline{\mathtt{f(a,b,c)}} \hookrightarrow_{\mathsf{i}} \mathtt{f(}\underline{\mathtt{c}}\mathtt{,c,c)} \hookrightarrow_{\mathsf{i}} \mathtt{f(a,}\underline{\mathtt{c}}\mathtt{,c)} \hookrightarrow_{\mathsf{i}} \underline{\mathtt{f(a,b,c)}} \hookrightarrow_{\mathsf{i}} \cdots$$

Note that the first rule of $\mathcal{R}$ is not μ-conservative now.

9.1 Relaxing Monotonicity with CS-DPs

Fernández's criterion was also adapted to deal with proofs of termination of rewriting using dependency pairs, what allows us using reduction pairs instead of reduction orderings in proofs of termination.

In previous sections, we have shown how to prove innermost termination of *CSR* by using ICSDPs. Now, we can adapt the use of CS-usable arguments to be applied in proofs of innermost μ-termination with ICSDPs. We do that by providing a new processor for dealing with innermost μ-termination problems.

Theorem 13 *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $\mu_{\mathcal{A}}(f) = \mathbf{UA}_{\mu}(f, \mathcal{R}, \mathcal{P})$ for all $f \in \mathcal{F} \cup \mathcal{G} \cup \mathcal{H}$ and $(\gtrsim, \sqsupset)$ be a $\mu_{\mathcal{A}}$-reduction pair such that*

1. *$\mathcal{P}$ and $\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P})$ are μ-conservative,*
2. *$\mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}) \subseteq \gtrsim$ and $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$,*

Let $\mathcal{P}_{\sqsupset} = \{u \to v \in \mathcal{P} \mid u \sqsupset v\}$. Then, the processor Proc_{Fer} given by

$$\mathsf{Proc}_{Fer}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}), \varnothing, \mu_{\mathcal{A}}, \mathbf{i})\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is sound.

Proof.

We have to prove that every infinite minimal innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain introduces an infinite minimal innermost $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}), \mathcal{S}, \mu_{\mathcal{A}}, \mathbf{i})$-chain. We proceed by contradiction. Assume that there is an infinite minimal innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain A, but that there is no infinite minimal innermost $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathbf{U}_0(\mathcal{R}, \mu, \mathcal{P}), \mathcal{S}, \mu_{\mathcal{A}}, \mathbf{i})$-chain. Due to the finiteness of $\mathcal{P}$, and since $\mathcal{P}$ is conservative, we have $\mathcal{P}_{\mathcal{X}} = \varnothing$. Thus, we can assume that there is $\mathcal{Q} \subseteq \mathcal{P}$ such that A has a tail B

$$\sigma(u_1) \hookrightarrow_{\mathcal{Q},\mu} t_1 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_2) \hookrightarrow_{\mathcal{Q},\mu} t_2 \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_3) \hookrightarrow_{\mathcal{Q},\mu} \cdots$$

for some substitution σ, where all pairs in $\mathcal{Q}$ are infinitely often used, and, for all $i \geq 1$, since all $u_i \to v_i \in \mathcal{P}$ are conservative $u_i \to v_i \in \mathcal{Q}_{\mathcal{G}}$ (i.e. $\mathcal{P}_{\mathcal{X}} = \mathcal{Q}_{\mathcal{X}} = \varnothing$), then $t_i = \sigma(v_i)$ such that for all $i > 0$, $\sigma(u_i)$ is argument μ-normalized and $\sigma(v_i)$ is innermost $(\mathcal{R}, \mu)$-terminating. By Proposition 9 and

11, every innermost step in the sequence $t_i \overset{!}{\hookrightarrow}_{\mathcal{R},\mu,\mathsf{i}} \sigma(u_{i+1})$ is performed at a $\mu_{\mathcal{A}}$-replacing position by means of a conservative rule in $\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})$:

$$\sigma(u_1) \hookrightarrow_{\mathcal{Q}_{\mathcal{G}},\mu_{\mathcal{A}}} t_1 \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}},\mathsf{i}} \sigma(u_2) \hookrightarrow_{\mathcal{Q}_{\mathcal{G}},\mu_{\mathcal{A}}} t_2 \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}},\mathsf{i}} \sigma(u_3) \hookrightarrow \cdots$$

Since $u_i \ (\gtrsim \cup \sqsupset) \ v_i$ for all $u_i \to v_i \in \mathcal{Q} \subseteq \mathcal{P}$, by stability of $\gtrsim$ and $\sqsupset$, we have $\sigma(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma(v_i)$ for all $i \geq 1$.

No pair $u \to v \in \mathcal{Q}$ satisfies that $u \sqsupset v$. Otherwise, we get a contradiction by considering that since all pairs $\in \mathcal{P}$ are conservative $u_i \to v_i \in \mathcal{Q}_{\mathcal{G}}$, then $t_i = \sigma(v_i) \overset{!}{\hookrightarrow}_{\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}},\mathsf{i}} \sigma(u_{i+1})$ and $t_i \gtrsim \sigma(u_{i+1})$. Since we have $\sigma(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma(v_i) = \sigma(v_i) = t_i$, by using transitivity of $\gtrsim$ and compatibility between $\gtrsim$ and $\sqsupset$, we conclude that $\sigma(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma(u_{i+1})$. Since $u \to v$ occurs infinitely often in B, there is an infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that $\sigma(u_i) \sqsupset \sigma(u_{i+1})$ for all $i \in \mathcal{I}$. And we have $\sigma(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma(u_{i+1})$ for all other $u_i \to v_i \in \mathcal{Q}$. Thus, by using the compatibility conditions of the μ-reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence which contradicts well-foundedness of $\sqsupset$. Therefore, $\mathcal{Q} \subseteq (\mathcal{P} \setminus \mathcal{P}_{\sqsupset})$. Since $\mu_{\mathcal{A}} \subseteq \mu$ and $\mathsf{NF}_{\mu_{\mathcal{A}}}(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P})) \supseteq \mathsf{NF}_{\mu}(\mathcal{R})$, we have that $\sigma(u_i) \in \mathsf{NF}_{\mu_{\mathcal{A}}}(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}))$. By Proposition 9, innermost $(\mathcal{R},\mu)$-termination of $\sigma(v_i)$ implies innermost $(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu)$- termination of $\sigma(v_i)$ for all $i \geq 1$ and by Proposition 11, innermost $(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu)$-termination of $\sigma(v_i)$ implies innermost $(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}})$-termination, so we get that innermost $(\mathcal{R},\mu)$-termination of $\sigma(v_i)$ implies innermost $(\mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}})$-termination. Hence, B is an infinite minimal innermost $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathbf{U}_0(\mathcal{R},\mu,\mathcal{P}),\mu_{\mathcal{A}})$-chain, thus leading to a contradiction.

□

Corollary 4 can be generalized to (certain) non-μ-conservative CS-TRSs thanks to Theorem 13. Now, for a given CS-TRS $(\mathcal{R},\mu)$ that satisfies the conditions of Theorem 13, we can prove its innermost μ-termination by relaxing μ-monotonicity requirements for each cycle.

10 Narrowing Transformation

Although, function TCap provides a good approximation of the graph, it can lead to *overestimate* the arcs that connect two dependency pairs. As already observed by Arts and Giesl for the standard and innermost case [AG00], in our setting the overestimation comes when a (noncollapsing) pair $u_i \to v_i$ is followed in a chain by a second one $u_{i+1} \to v_{i+1}$ and v_i and u_{i+1} are not directly unifiable, i.e., at least one innermost μ-rewriting step is needed to innermost μ-reduce $\sigma(v_i)$ to $\sigma(u_{i+1})$. Then, the innermost μ-reduction from $\sigma(v_i)$ to $\sigma(u_{i+1})$ requires at least one step, i.e., we always have $\sigma(v_i) \hookrightarrow_{\mathcal{R},\mu^\sharp} \sigma(v_i') \hookrightarrow^*_{\mathcal{R},\mu^\sharp} \sigma(u_{i+1})$. Furthermore, we could discover that v_i has *no μ-narrowings*. In this case, we know that no innermost chain starts from $\sigma(v_i)$. A restriction that have to be taken into account when μ-narrowing a noncollapsing pair $u \to v$ is that the

μ-replacing variables in v have to be μ-replacing in u as well (this corresponds with the notion of conservativeness), but furthermore, they cannot be both μ-replacing and non-μ-replacing at the same time. This corresponds to the following definition.

Definition 19 (Strongly Conservative [GLU08]**)** *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. A rule $l \to r$ is strongly μ-conservative if it is μ-conservative and $\mathcal{V}ar^{\mu}(l) \cap \mathcal{V}ar^{\not\mu}(l) = \mathcal{V}ar^{\mu}(r) \cap \mathcal{V}ar^{\not\mu}(r) = \varnothing$.*

In [AGL10], we define the μ-narrowing processor. Of course, μ-narrowing can also be used in proofs of innermost termination of *CSR*. In the standard setting, when using narrowing for proving innermost termination we do not require that the right-hand side of the dependency pair to be narrowed is linear since the involved substitution σ is normalized. However, in the context-sensitive setting, if the pair to be μ-narrowed is *not strongly μ-conservative*, we can not ensure that the variables on the right-hand side are μ-normalized so *we also have to demand linearity*. When dealing with innermost narrowing in context-sensitive rewriting we can drop the linearity condition if the pair to be μ-narrowed is strongly conservative since all μ-replacing variables in the right-hand side of a pair are instantiated to μ-normal form and μ-reductions can not take place on them.

Theorem 14 (Innermost Narrowing processor) *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$ be a CS problem. Let $u \to v \in \mathcal{P}$ be such that*

1. *for all $u' \to v' \in \mathcal{P}$ (with possibly renamed variables), v and u' do not unify or they unify by some mgu θ such that one of the terms $\theta(u)$ or $\theta(u')$ is not a μ-normal form.*

Let $\mathcal{Q} = (\mathcal{P} - \{u \to v\}) \cup \{u' \to v' \mid u' \to v' \text{ is a } \mu\text{-narrowing of } u \to v\}$. Then, the processor Proc_{Inarr} given by

$$\mathsf{Proc}_{Inarr}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i}) = \begin{cases} \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{if (1) holds} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})\} & \text{otherwise} \end{cases}$$

is

1. *sound whenever $u \to v$ is strongly conservative, and*
2. *complete in all cases.*

Proof.

We have to prove that there is an infinite minimal innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain iff there is an infinite minimal innermost $(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain. We prove that for every minimal innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$", there is an innermost μ-narrowing v' of v with the mgu θ such that "$\ldots, u_1 \to v_1, \theta(u) \to v', u_2 \to v_2, \ldots$" is also a minimal innermost $(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain.

If "$\ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots$" is a minimal innermost $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu, \mathbf{i})$-chain, then there is an substitution σ such that for all pairs $s \to t$ in the chain,

1. if $s \to t \in \mathcal{P}_\mathcal{G}$, then $\sigma(t)$ is μ-terminating and it μ-reduces innermost to the instantiated left-hand side $\sigma(s')$ of the next pair $s' \to t'$ in the chain

2. if $s \to t = s \to x \in \mathcal{P}_\mathcal{X}$, then $\sigma(x) \overset{\Lambda}{\hookrightarrow}^*_{\mathcal{S}_{\triangleright_\mu},\mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}_\sharp,\mu} t$ and t, which is innermost μ-terminating, μ-reduces innermost to the instantiated left-hand side $\sigma(s')$ of the next pair $s' \to t'$ in the chain.

3. all instantiated left-hand sides are μ-normal forms w.r.t. $(\mathcal{R}, \mu)$.

Assume that σ is a substitution satisfying the above requirements and such that the length of the sequence $\sigma(v) \hookrightarrow^*_{\mathcal{R},\mu,\mathrm{i}} \sigma(u_2)$ is *minimal*.

Note that $\sigma(v) \neq \sigma(u_2)$. Otherwise σ would unify v and u_2, where both, u and v_2 are μ-normal forms, hence, there is a term q such that $\sigma(v) \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} q \hookrightarrow^*_{\mathcal{R},\mu,\mathrm{i}} \sigma(u_2)$.

The reduction $\sigma(v) \hookrightarrow_{\mathcal{R},\mu,\mathrm{i}} q$ cannot take place within a binding of σ because $u \to v$ is strongly conservative. Hence, $\sigma(u)$ would not be a μ-normal form which violates the last condition for σ. In the innermost case, we do not have to demand linearity since μ-replacing variables in v come from being replacing in u (strongly conservative) and they are instantiated to μ-normal forms and no one can be reduced in v. The remainder of the proof is completely analogous to the noninnermost case.

□

Example 20 *Consider the following example:*

$$\begin{array}{lcl} \mathtt{f}(\mathtt{s}(x)) & \to & \mathtt{f}(\mathtt{p}(\mathtt{s}(x))) \\ \mathtt{p}(\mathtt{s}(x)) & \to & x \end{array}$$

together with $\mu(\mathtt{f}) = \mu(\mathtt{s}) = \{1\}$ *and* $\mu(\mathtt{p}) = \varnothing$.

The only ICSDP that could generate a cycle is $\mathtt{F}(\mathtt{s}(x)) \to \mathtt{F}(\mathtt{p}(\mathtt{s}(x)))$. *However since the right-hand side* $\mathtt{F}(\mathtt{p}(\mathtt{s}(x)))$ *does not unify with any (renamed) the left-hand side (including itself) and the pair is strongly conservative, we can apply μ-narrowing. Therefore, the pair can be μ-narrowed at position 1 (notice that* $\mu(f) = \mu(\mathtt{F}) = \{1\}$*) by using the rule* $\mathtt{p}(\mathtt{s}(x)) \to x$. *Then, the pair is transformed into the pair* $\mathtt{F}(\mathtt{s}(y)) \to \mathtt{F}(y)$ *that can be easy disregarded by using the subterm criterion*[1].

11 Experiments

We have implemented the techniques described in the previous sections as part of the tool MU-TERM [AGL+10]. In order to evaluate the techniques which are reported in this paper we have made some benchmarks. We have considered the examples in the Termination Problem Data Base[2] (TPDB).

[1] Instead of using in the proof a polynomial interpretation with rationals, like MU-TERM or matrix interpretations like AProVE.

[2] `http://www.termination-portal.org/wiki/TPDB`

	ICSDPs	Transformations
YES score	95/109	60/109
YES average time	0.7 sec.	1.5 sec.

Table 1: Comparative in proofs of termination of innermost *CSR*

	C	GM	iGM
YES score	33	57	42

Table 2: Comparing transformations for proving termination of innermost *CSR*

11.1 Proving Termination of Innermost *CSR*: Direct Techniques vs. Transformations

Although there is no special TPDB category for innermost termination of *CSR* (yet) we have used the examples used in the *CSR*category in order to test our techniques for proving termination of innermost *CSR*. The TPDB v7.0.2 contains 109 examples of CS-TRSs. In order to evaluate our direct techniques in comparison with the transformational approach of [GM02b, GM04, Luc01a], where termination of innermost *CSR* for a CS-TRS ($\mathcal{R}$, μ) is proved by proving innermost termination of a transformed TRS $\mathcal{R}^{\mu}_{\Theta}$, where Θ specifies a particular transformation (see [GM02a, GM02b] for a survey on this topic), we have transformed the set of examples by using the transformations that are correct for proving innermost termination of *CSR*: Giesl and Middeldorp's correct transformations for proving termination of innermost *CSR*, see [GM02b], although we use the 'authors-based' notation introduced in [Luc06]: GM and C for transformations 1 and 2 for proving termination of *CSR* introduced in [GM04], and iGM for the specific transformation for proving termination of innermost *CSR* introduced in [GM02b]. Then we have proved innermost termination of the set of examples with AProVE [GST06], which is able to prove innermost termination of standard rewriting. The results are summarized in Table 1 and 11.1. Further details can be found here:

`http://www.dsic.upv.es/~balarcon/iCSR/benchmarks.html`

These are the first known benchmarks comparing not only transformational techniques vs. direct (DP-based) techniques, but also the existing correct transformations for proving innermost termination of *CSR* among them. They show that, quite surprisingly, the iGM transformation (which is in principle the more suitable one for proving innermost termination of *CSR*) obtains worse results than GM (in the average).

In [AL07], we obtained 70 over 90 successful proofs against 44 for transformations (it was used the TPDB v3.2). Obviously, the use of ICSDPs were imposed without doubts for proving innermost termination of *CSR*. Moreover, now, with the recent developments of MU-TERM embracing the DP-framework, MU-TERM would solve 77 over those 90 of the previous version (and 18 over

the new 19 included in the last one). Therefore, from the results in Table 1, it is clear that using transformations for proving termination of innermost *CSR* makes no sense after introducing the ICSDP framework.

11.2 Proving Innermost Termination of *CSR*: Relaxing Monotonicity Requirements

For our experiments about proving termination of innermost *CSR* by means of a new replacement map which imposes less monotonicity requirements we have used the set of examples mentioned in Section 11.1.

We have implemented the use of Theorem 13 to deal with nonconservative systems (but conservative cycles). MU-TERM tries to solve each μ-conservative cycle (with associated μ-conservative usable rules) by using CS-usable arguments as the new replacement map. This implementation of MU-TERM succeed over the same 95 examples, the same number of examples that we had already solved using ICSDPs. The time average rates has no exhibit substantial differences. Further details can be found here:

`http://www.dsic.upv.es/~balarcon/iCSR_UA/benchmarks.html`

Although no improvement over the practical use of ICSDPs explained in previous subsection is shown, we expect that in the future, when we implement nonmonotonic orderings in our termination tool MU-TERM we take advantage of this technique.

Moreover, we have implemented the use of Corollary 11 in [Fer05] to prove innermost termination of TRSs by proving μ-termination of the CS-TRS obtained after using the usable arguments as replacement map (this was one of the main results in Fernández's paper). The relevance of this result in practice had not been tested yet as no implementation of Fernández's results was available (to our knowledge). In order to evaluate it, we have considered the examples used in the *innermost category*. There are 358 examples. Using usable arguments (we call this tool MU-TERM UA), MU-TERM succeeds in 158 examples. However, we have also implemented the use of (standard) dependency pairs for proving innermost termination (according to [AG00, Theorem 37]) together with the narrowing refinement (we call this tool MU-TERM iDPs) and we are able to prove 199 examples, including all examples solved with Fernández's criterion.

Therefore, it seems that using her result to prove innermost termination of rewriting is not as good idea (at least with the considered set of examples) since we loose some examples and the average time is worse. The results are summarized in Table 3. Further details can be found in:

`http://www.dsic.upv.es/~balarcon/UA/benchmarks.html`

All this shows that we do not obtain any real improvement over the basic technique of dependency pairs for proving innermost termination at least for the set of considered examples.

	MU-TERM UA	MU-TERM iDPs
YES score	158	199
YES average time	4.87 sec.	3.31 sec.

Table 3: Benchmarks for innermost termination of rewriting

11.3 Transforming CS-dependency Pairs

We have also implemented innermost μ-narrowing in MU-TERM. Due to the possibility of performing an unbounded number of narrowing steps, the μ-narrowing transformation could be infinite (this also happens in the standard approach). In order to implement the transformation, we have chosen to use *one-step μ-narrowing* only if the innermost context-sensitive dependency graph obtained has less cycles and arcs than the original one. One of the best advantages of using μ-narrowing lies in the possibility of dismissing some CS-DPs if the right-hand sides do not unify with any left-hand side of another (possible renamed) CS-DP and they have no μ-narrowings.

11.4 Termination Competition

Thanks to the new developments reported in this paper and in [AGL10, GL10], MU-TERM 5.07 has proven to be the most powerful tool for proving termination of *CSR* in the *context-sensitive* subcategory of the 2007, 2009 and 2010 editions of the International Competition of Termination Tools[3].As we have commented, under some conditions, termination of *CSR* and termination of innermost *CSR* coincide [GM02b, GL02b]. For this reason, one of the most important aspect of innermost *CSR* is its use for proving termination of *CSR* as part of the CSDP framework. We switch from termination of *CSR* to termination of innermost *CSR*whenever termination is equivalent, for which we can apply the existing processors more successfully. Actually, we proceed like that in $30-50\%$ of the *CSR* termination problems which are proved by MU-TERM 5.0.

12 Conclusions

The results of this paper are revised and extended versions of the results published in [AL07, AL09], having into account all improvements made in the CSDP framework in [AGL10, GL10].

[3]See `http://www.lri.fr/~marche/termination-competition/2007/`, where only AProVE and MU-TERM participated, and `http://termcomp.uibk.ac.at/termcomp/` where there were three more tools in the competition: AProVE, Jambox [End], and VMTL [SG09]. AProVE and MU-TERM solved the same number of examples but MU-TERM was much faster. The same situation has happened in 2010 (but without Jambox's participation).

12.1 Theoretical Contributions

We have investigated the structure of infinite innermost context-sensitive rewrite sequences starting from (strongly) minimal innermost non-μ-terminating terms (Theorem 1). This knowledge has been used to provide an appropriate definition of innermost context-sensitive dependency pair (Definition 5), and the related notion of innermost chain (Definition 8). We have proved that it can be used to characterize innermost μ-termination (Theorems 2 and 3). We have provided a suitable adaptation of Giesl et al.'s *dependency pair framework* to innermost *CSR* by defining appropriate notions of *CS problem* (Definition 9) and *CS processor* (Definition 10). In this setting we have described a number of sound and (most of them) complete CS-processors which can be used in any practical implementation of the ICSDP framework. In particular, we have introduced the notion of (estimated) *innermost context-sensitive (dependency) graph* (Definitions 12 and 15) by using functions to approximate it (Definition 14) and the associated CS processor showing how to automatically prove innermost μ-termination by means of the ICS dependency graph (Theorem 10). We have formulated the notion of basic usable rules showing how to use them in proofs of innermost termination of *CSR* (Definition 17, Theorem 11). Narrowing context-sensitive dependency pairs has also been investigated. It can also be helpful to simplify or restructure the dependency graph and eventually simplify the proof of (innermost) termination (Theorem 14). We have also shown how to relax monotonicity requirements for proving innermost termination of context-sensitive rewriting. We have adapted Fernández's approach [Fer05] to be used for proving innermost termination of context-sensitive rewriting (Theorems 12 and 13).

12.2 Applications and Practical Impact

We have implemented these ideas as part of the termination tool MU-TERM [AGIL07, Luc04]. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of *CSR*. Actually, ICSDPs were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007, 2009 and 2010 competitions of termination tools.

Up to our contributions, no direct method has been proposed to prove termination of innermost *CSR*. So this is the first proposal of a direct method for proving termination of innermost *CSR*. We have extended the DP framework [GTS04, GTSF06] to prove innermost termination of TRSs to innermost *CSR* (thus also extending [AGL07]). Our benchmarks show that the use of ICSDPs dramatically improves the performance of existing (transformational) methods for proving termination of innermost *CSR*.

12.3 Future Work

As remarked in the introduction, we aim at applying all previous developments to deal with termination of Maude programs. Since its computational mechanism can be thought of as kind of "context-sensitive call by value", we believe that our research is a essential contribution to the development of tools for proving termination of Maude programs.

References

[AAC+08] E. Albert, P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini. Termination Analysis of Java Bytecode. FMOODS 2–18. 2008.

[Ala08] B. Alarcón. Innermost Termination of Context-Sensitive Rewriting. Master Thesis, DSIC, Universidad Politécnica de Valencia. December 2008.

[AEF+08] B. Alarcón, F.Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving context-sensitive dependency pairs. In I. Cervesato, H. Veith and A. Voronkov, editors, *Proc. of 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, LNAI 5330:636–651, Springer-Verlag, 2008.

[AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.

[AGIL07] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.

[AGL06] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, LNCS 4337:297–308, Springer-Verlag, 2006.

[AGL07] B. Alarcón, R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.

[AGL10] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-sensitive dependency pairs. Information and Computation 208(8), 922–968, 2010.

[AGL+10] B. Alarcón, R. Gutiérrez, S. Lucas and R. Navarro-Marset. Proving Termination Properties with MU-TERM. *Proc. of Thirteenth*

International Conference on Algebraic Methodology And Software Technology, AMAST2010, to appear 2010.

[AL07] B. Alarcón and S. Lucas. Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In B. Konev and F. Wolter, editors, *Proc. of 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, LNAI 4720:73–87, Springer-Verlag, 2007.

[AL09] B. Alarcón, S. Lucas.
Using Context-Sensitive Rewriting for Proving Innermost Termination of Rewriting. Electronic Notes in Theoretical Computer Science, 248:3–17, 2009.

[AM93] G. Aguzzi and U. Modigliani. Proving termination of logic programs by transforming them into equivalent term rewriting systems. *Proc. of the 13th Conference on Foundations of Software Technology and Theoretical Computer Science,FST&TCS93*, LNCS 761: 114–124, Springer-Verlag, 1993.

[BM06] R. Bruni and J. Meseguer. Semantic foundations for generalized rewrite theories. *Theoretical Computer Science* 351(1):386–414, 2006.

[BMS05] A. R. Bradley, Z. Manna, and H. B. Sipma. Termination of polynomial programs. *Proc. of the 6th International Conference on Verification, Model Checking, and Abstract Interpretation, VMCAI 05*, LNCS, 3385:113–129, Springer-Verlag, 2005.

[BN98] F. Baader and T. Nipkow. Term Rewriting and All That. Cambridge University Press, 1998.

[Bor03] C. Borralleras. Ordering-based methods for proving termination automatically. PhD Thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, May 2003.

[CDE+07] Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. All About Maude – A High-Performance Logical Framework. LNCS 4350, Springer-Verlag, 2007.

[CLS05] M. Codish, V. Lagoon, and P. Stuckey. Testing for termination with monotonicity constraints. *Proc. of the 21th International Conference on Logic Programming, ICLP 05*, LNCS 3668: 326–340, Springer-Verlag, 2005.

[CPR06] B. Cook, A. Podelski, and A. Rybalchenko. Termination proofs for systems code. *Proc. of the ACM SIGPLAN Conference on Programming Language Design and Implementation, PLDI 06*, pages 415–426. ACM Press, 2006.

[CS02] M. Colon and H. B. Sipma. Practical methods for proving program termination. *Proc. of the 14th International Conference on Computer Aided Verification, CAV 02*, LNCS 2034: 442–454, Springer-Verlag, 2002.

[CT99] M. Codish and C. Taboch. A semantic basis for the termination analysis of logic programs. *The Journal of Logic Programming*, 41(1):103–123, 1999.

[DD94] D. De Schreye and S. Decorte. Termination of logic programs: The neverending story. *Journal of Logic Programming*, 19/20:199–260, 1994.

[Der04] N. Dershowitz. Termination by Abstraction. In B. Demoen and V. Lifschitz, editors, *Proc. of 20th International Conference on Logic Programming, ICLP'04*, LNCS 3132:1–18, Springer-Verlag, 2004.

[DL01] N. Dershowitz, N. Lindenstrauss, Y. Sagiv, and A. Serebrenik. A general framework for automatic termination analysis of logic programs. *Appl. Algebra Eng. Commun. Comput.*, 12(1/2):117–156, 2001.

[DLM+04] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Termination of Membership Equational Programs. In *Proc. of 2004 ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147–158, Verona, Italy. ACM Press, 2004.

[DLM+08] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, 21(1-2):59–88, 2008.

[DS02] D. De Schreye and A. Serebrenik. Acceptability with general orderings. *Computational Logic: Logic Programming and Beyond*, LNCS 2407: 187–210, Springer-Verlag, 2002.

[Emm08] F. Emmes. Automated Termination Analysis of Context-Sensitive Term Rewrite Systems. Diplomarbeit im Studiengang Informatik, RWTH Aachen, 2008.

[End] Endrullis, J.: Jambox, Automated Termination Proofs for String and Term Rewriting. Available at `http://joerg.endrullis.de/jambox.html`

[Fer05] M. L. Fernández. Relaxing monotonicity for innermost termination. Information Processing Letters 93(3):117–123, 2005.

[FGJM85] K. Futatsugi, J. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85*, pages 52–66, ACM Press, 1985.

[FN97] K. Futatsugi and A. Nakagawa. An Overview of CAFE Specification Environment – An algebraic approach for creating, verifying, and maintaining formal specification over networks –. In *Proc. of 1st International Conference on Formal Engineering Methods*, 1997.

[GAO02] J. Giesl, T. Arts, and E. Ohlebusch. Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.

[Gie95] J. Giesl. Termination analysis for functional programs using term orderings. *Proc. of the 2nd International Static Analysis Symposium, SAS 95*, LNCS 983:154–171, Springer-Verlag, 1995.

[GL02a] B. Gramlich and S. Lucas. Simple Termination of Context-Sensitive Rewriting. In B. Fischer and E. Visser, editors, *Proc. of III ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02*, pages 29–41. ACM Press, 2002.

[GL02b] B. Gramlich and S. Lucas. Modular Termination of Context-Sensitive Rewriting. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'02*, pages 50–61. ACM Press, 2002.

[GL10] R. Gutiérrez, S. Lucas. Proving Termination in the Context-Sensitive Dependency Pair Framework. *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*. LNCS, 6381:19–35, Springer-Verlag, 2010.

[GLU08] R. Gutiérrez, S. Lucas, and X. Urbain. Usable Rules for Context-Sensitive Rewrite Systems. In A. Voronkov, editor, *Proc. of the 19th International Conference on Rewriting Techniques and Applications, RTA'08*, LNCS, 5117:126–141, Springer-Verlag, 2008.

[GM02a] J. Giesl and A. Middeldorp. Innermost termination of context-sensitive rewriting. Aachener Informatik-Berichte (AIBs), RWTH Aachen, 2002. http://citeseer.ist.psu.edu/giesl02innermost.html

[GM02b] J. Giesl and A. Middeldorp. Innermost termination of context-sensitive rewriting. In M. Ito and M. Toyama, editors, *Proc. of 6th International Conference on Developments in Language Theory, DLT'02*, LNCS 2450:231–244, Springer-Verlag, 2003.

[GM04] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.

[Gra95] B. Gramlich. Abstract Relations between Restricted Termination and Confluence Properties of Rewrite Systems, *Fundamenta Informaticae*, 24:3–23, 1995.

[Gra96] B. Gramlich. On Proving Termination by Innermost Termination. *Proc. 7th Int. Conf. on Rewriting Techniques and Applications Proc. of RTA'96*, LNCS 1103:93–107, Springer-Verlag, 1996.

[GSST06] J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated termination analysis for Haskell: From term rewriting to programming languages. *Proc. of the 17th International Conference on Rewriting Techniques and Applications, RTA 06*, LNCS 4098: 297–312, Springer-Verlag 2006.

[GST06] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281–286, Springer-Verlag, 2006. Available at `http://www-i2.informatik.rwth-aachen.de/AProVE`.

[GTS04] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors, *Proc. of XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04*, LNAI 3452:301–331, Springer-Verlag, 2004.

[GTS05] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and Disproving Termination of Higher-Order Functions. In B. Gramlich, editor, *Proc. of 5th International Workshop on Frontiers of Combining Systems, FroCoS'05*, LNAI 3717:216–231, Springer-Verlag, 2005.

[GTSF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006.

[GWM+00] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J.-P. Jouannaud. Introducing OBJ. In J. Goguen and G. Malcolm, editors, *Software Engineering with OBJ: algebraic specification in action*, Kluwer, 2000.

[HM04] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In V. van Oostrom, editor, *Proc. of XV International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091: 249–268, Springer-Verlag, 2004.

[HM05] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. *Information and Computation*, 199:172–199, 2005.

[HM07] N. Hirokawa and A. Middeldorp. Tyrolean termination tool: Techniques and features. *Information and Computation*, 205:474-511, 2007.

[HPW92] P. Hudak, S.J. Peyton-Jones, and P. Wadler. Report on the Functional Programming Language Haskell: a non–strict, purely functional language. *Sigplan Notices*, 27(5):1–164, 1992.

[KKS98] M. R. K. Krishna Rao, D. Kapur, and R. Shyamasundar. Transformational methodology for proving termination of logic programs. *Journal of Logic Programming*, 34(1):1–42, 1998.

[LJB01] C. S. Lee, N. D. Jones, and A. M. Ben-Amram. The size-change principle for program termination. *ACM Symposium on Principles of Programming Languages, POPL 01*, pages 81–92, ACM Press, 2001.

[LMS03] V. Lagoon, F. Mesnard, and P. J. Stuckey. Termination analysis with types is more accurate. *Proc. of the 19th International Conference on Logic Programming, ICLP 03*, LNCS 2916: 254–268, Springer-Verlag, 2003.

[Luc96] S. Lucas. Termination of context-sensitive rewriting by rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of 23rd. International Colloquium on Automata, Languages and Programming, ICALP'96*, LNCS 1099:122-133, Springer-Verlag, 1996.

[Luc98] S. Lucas. Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, 1998.

[Luc01a] S. Lucas. Termination of Rewriting With Strategy Annotations. In R. Nieuwenhuis and A. Voronkov, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, LNAI 2250:669–684, Springer-Verlag, 2001.

[Luc01b] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proc. of 3rd International Conference on Principles and Practice of Declarative Programming, Proc. of PPDP'01*, pages 82–93, ACM Press, 2001.

[Luc02] S. Lucas. Context-Sensitive Rewriting Strategies. *Information and Computation*, 178(1):293–343, 2002.

[Luc04] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of XV International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200–209, Springer-Verlag, 2004. Available at `http://zenon.dsic.upv.es/muterm`.

[Luc06] S. Lucas. Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation*, 204(12):1782–1846, 2006.

[LM08] S. Lucas and J. Meseguer. Order-Sorted Dependency Pairs. In S. Antoy, editor, *Proc. of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08*, pages 108–119, ACM Press, 2008.

[LM09] S. Lucas and J. Meseguer. Operational Termination of Membership Equational Programs: the Order-Sorted Way. In G. Rosu, editor, *Proc. of the 7th International Workshop on Rewriting Logic and its Applications, WRLA'08*, *Electronic Notes in Theoretical Computer Science*, 238(3):207–225, 2009.

[Mar94] M. Marchiori. Logic programs as term rewriting systems. *Proc. of the 4th International Conference on Algebraic and Logic Programming, ALP94*, LNCS 850:223–241, Springer-Verlag, 1994.

[Mar96] M. Marchiori. Proving existential termination of normal logic programs. *Proc. of the 5th International Conference on Algebraic Methodology and Software Technology, AMAST 96*, LNCS 1101: 375–390, Springer-Verlag, 1996.

[McC60] J. McCarthy. Recursive Functions of Symbolic Expressions and their Computations by Machine, Part I. *Communications of the ACM*, 3(4):184–195, 1960.

[Mid01] A. Middeldorp. Approximating dependency graphs using tree automata techniques. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of the International Joint Conference on Automated Reasoning, IJCAR'01*, LNAI 2083:593–610, Springer-Verlag, 2001.

[Mid02] A. Middeldorp, Approximations for strategies and termination *Electronic Notes in Computer Science*, volume 70(6), 2002.

[NSEP92] E.G.J.M.H. Nöcker, J.E.W. Smetsers, M.C.J.D. van Eekelen, and M.J. Plasmeijer. Concurrent Clean. In E.H.L. Aarts, J. Leeuwen, and M. Rem, editors, *Proc. of Parallel Architectures and Languages Europe, PARLE'91*, LNCS 506:202–219, Springer-Verlag, 1992.

[OBEG10] C. Otto, M. Brockschmidt, C. von Essen and J. Giesl Automated Termination Analysis of Java Bytecode by Term Rewriting. *Proc of 21st International Conference on Rewriting Techniques and Applications, RTA 2010*, pages 259–276, 2010.

[Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting.* Springer-Verlag, 2002.

[SG09] Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09.* LNCS, 5595:285–294. Springer-Verlag, 2009.

[SGN09] P. Schneider-Kamp, J. Giesl and M. T. Nguyen. The Dependency Triple Framework for Termination of Logic Programs. *Proc. of the 19th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 09*, LNCS 6037: 37–51, Springer-Verlag, 2009.

[SGST06] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated termination analysis for logic programs by term rewriting. *Proc. of the 16th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 06*, LNCS 4407: 177–193, Springer-Verlag, 2006.

[Sma04] J.-G. Smaus. Termination of logic programs using various dynamic selection rules. *Proc. of the 20th International Conference on Logic Programming, ICLP 04*, LNCS 3132: 43–57, Springer-Verlag, 2004.

[TeR03] TeReSe. *Term Rewriting Systems*. Cambridge University Press, 2003.

[Thi07] R. Thiemann. The DP Framework for Proving Termination of Term Rewriting. PhD Thesis. Available as Technical Report AIB-2007-17, RWTH Aachen University, Germany, 2007.

[Tiw04] A. Tiwari. Termination of linear programs. *Proc. of the 16th International Conference on Computer Aided Verification, CAV 04*, LNAI, 3097:70–82, Springer-Verlag, 2004.

[Xi02] H. Xi. Dependent types for program termination verification. *Higher-Order and Symbolic Computation*, 15(1):91–131, 2002.

[Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of VII International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172–186, Springer-Verlag, 1997.

18.11 A Dependency Pair Framework for $A \vee C$-Termination

11. B. Alarcón, R. Gutiérrez, S. Lucas, and J. Meseguer. **A Dependency Pair Framework for $A \vee C$-Termination**. Technical Report, http://hdl.handle.net/10251/10797, DSIC-ELP, UPV, 2011.

A Dependency Pair Framework for $A \vee C$-Termination

Beatriz Alarcón Raúl Gutiérrez Salvador Lucas
José Meseguer*
DSIC, Universidad Politécnica de Valencia
Camino de Vera s/n, E-46022 Valencia, Spain
{balarcon ,rgutierrez ,slucas}@dsic.upv.es
*CS Dept. University of Illinois at Urbana-Champaign, IL, USA
meseguer@uiuc.edu

Abstract

The development of powerful techniques for proving termination of rewriting modulo a set of equational axioms is essential when dealing with rewriting logic-based programming languages like CafeOBJ, Maude, ELAN, OBJ, etc. One of the most important techniques for proving termination over a wide range of variants of rewriting (strategies) is the *dependency pair approach*. Several works have tried to adapt it to rewriting modulo *associative and commutative* (AC) equational theories, and even to more general theories. However, as we discuss in this paper, no appropriate notion of minimality (and minimal chain of dependency pairs) which is well-suited to develop a *dependency pair framework* has been proposed to date. In this paper we carefully analyze the structure of infinite rewrite sequences for rewrite theories whose equational part is *any combination* of associativity and/or commutativity axioms, which we call $A \vee C$*-rewrite theories*. Our analysis leads to a more accurate and optimized notion of dependency pairs through the new notion of *stably minimal term*. We then develop a suitable dependency pair framework for proving termination of $A \vee C$-rewrite theories.

1 Introduction

Rewriting with rules R modulo axioms E is a widely used technique in both rule-based programming languages and in automated deduction. Consequently, termination of rewriting modulo specific equational axioms E (e.g., associativity-commutativity, AC) has been studied. Methods for proving termination of rewriting systems modulo AC-axioms are known and even implemented. Several works have tried to adapt the *dependency pair approach* (DP-approach [1]) to rewriting modulo *associative and commutative* (AC) theories [20, 16, 17, 18, 21].

```
fmod LIST&SET is
  sorts Bool Nat List Set .
  subsorts Nat < List Set .
  ops true false : -> Bool .
  ops _and_ _or_ : Bool Bool -> Bool [assoc comm] .
  op 0 : -> Nat .
  op s_ : Nat -> Nat .
  op _;_ : List List -> List [assoc] .
  op null : -> Set .
  op __ : Set Set -> Set [assoc comm] .
  op _in_ : Nat Set -> Bool .
  op _==_ : List List -> Bool [comm] .
  op list2set : List -> Set .
  var  B : Bool .             vars N M : Nat .
  vars L L' : List .          var  S : Set .
  eq N N = N .
  eq true and B = B .         eq false and B = false .
  eq true or B = true .       eq false or B = B .
  eq 0 == s N = false .       eq s N == s M = N == M .
  eq N ; L == M = false .     eq N ; L == M ; L' = (N == M) and L == L' .
  eq L == L = true .
  eq list2set(N) = N .        eq list2set(N ; L) = N list2set(L) .
  eq N in null = false .      eq N in M S = (N == M) or N in S .
endfm
```

Figure 1: Example in Maude syntax [7]

The corresponding proof methods, though, cannot be applied to commonly occurring combinations of axioms that fall outside their scope.

Example 1 *Consider the (order-sorted) TRS specified in Maude in Figure 1. It has four sorts:* `Bool`*,* `Nat`*,* `List`*, and* `Set`*, with* `Nat` *included in both* `List` *and* `Set` *as a subsort. That is, a natural number n is simultaneously regarded as a list of length 1 and as a singleton set. The terms of each sort are, respectively, booleans, natural numbers (in Peano notation), lists of natural numbers, and finite sets of natural numbers. The rewrite rules in this module then define various functions such as* `_and_` *and* `_or_`*, a function* `list2set` *associating to each list its corresponding set, the set membership predicate* `_in_`*, and an equality predicate* `_==_` *on lists. Furthermore, the idempotency of set union is specified by the first equation. All these equations rewrite terms* modulo *the equational axioms declared in the module. Specifically,* `_and_` *and* `_or_` *have been declared associative and commutative with the* `assoc` *and* `comm` *keywords, the list concatenation operator* `_;_` *has been declared associative using the* `assoc` *keyword; the set union operator* `__` *has been declared associative, commutative using the* `assoc` *and* `comm`*, keywords; and the* `_==_` *equality predicate has been declared commutative using the* `comm` *keyword. The succinctness of this specification is precisely due to the power of rewriting modulo axioms, which typically uses considerably fewer rules that standard rewriting.*

Methods for proving termination of AC-theories could not be applied to prove termination of the TRS in Figure 1 (we would not care about sort information

here), where we have an *arbitrary combination* of associative and/or commutative axioms which we call an *$A\vee C$-rewrite theory* in this paper. Furthermore, to the best of our knowledge, the *Dependency Pair Framework* (DP-framework [13]), which is the basis of state-of-the-art tools for proving termination of (different variants of) term rewriting has not yet been adapted to the AC case.

In this paper, we address these two problems. Giesl and Kapur generalized the previous works on AC-termination with dependency pairs to deal with more general kinds of equational theories E satisfying some restrictions [10]. In principle, the $A\vee C$-theories that we investigate here fit the main outlines of Giesl and Kapur's approach. However, as we discuss below, the paper [10] did not provide any definition of *minimal chain*, which is needed for further developments in the DP-framework. In the DP-framework, the central notion regarding termination proofs is that of *DP problem*: the goal is checking the absence (or presence) of the so-called *infinite minimal chains*, where the notion of minimal chain can be thought of as an abstraction of the infinite rewrite sequences starting from *minimal non-terminating terms*. The most important notion regarding mechanization of the proofs is that of *processor*. A (correct) processor basically transforms DP problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. Here 'simpler' usually means that fewer pairs are involved. Processors are used in a pipe (more precisely, a *tree*) to incrementally simplify the original DP problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. This is the crucial new feature of the DP-framework w.r.t. the DP-approach of [1]. This makes it very powerful as a basis for implementing termination provers.

Before being able to extend the DP-framework to deal with $A\vee C$-theories, we start by giving a more refined notion of minimality. In fact, the notion of minimality which is used in [10] is the straightforward extension of the one which is used to prove termination of standard rewriting but without dealing with *E-equivalence preservation* which, as we show below, is essential to provide an appropriate notion of minimal E-nonterminating term for $A\vee C$-theories E which can be used to define a suitable $A\vee C$-DP-framework. We carefully analyze the structure of infinite rewrite sequences for $A\vee C$-rewrite theories. This leads to appropriate definitions of $A\vee C$-dependency pair and of minimal chain.

1.1 Plan of the paper

The results, techniques, and tools that derive from our work can be of interest to a fairly wide audience. The material in this paper will be more familiar, however, to specialists interested in termination and in proving termination of rewriting modulo equational theories. Throughout the paper we made a serious effort to provide sufficient intuition and informal descriptions for our main definitions and results. After some technical preliminaries, in Sections 2 and 3, the paper is structured in three main parts:

1. In Section 4 we investigate the drawbacks of previous notions of minimal term when modeling infinite $A \vee C$-rewrite sequences. Then, we introduce the notion of *stably minimal* E-nonterminating term, which is the basis of our development. Section 5 investigates the structure of infinite sequences starting from such stably minimal terms. This analysis is essential in order to provide an appropriate definition of $A \vee C$-dependency pair and the related notions of chain, graph, etc.

2. Section 6 uses these results to formalize our notions of $A \vee C$-dependency pairs and of minimal chains and shows how to use them to *characterize* termination of $A \vee C$-rewrite theories.

3. We describe a suitable framework for dealing with proofs of $A \vee C$-termination by using these results. Section 7 extends the *dependency pair framework* [12, 13] to $A \vee C$-termination by defining appropriate notions of $A \vee C$ *problem* and $A \vee C$ *processor* that rely on the results obtained in the second part of the paper. In particular, we introduce the notion of $A \vee C$-dependency graph and the associated $A \vee C$ processor. We also show how to use orderings for defining a second processor and other auxiliary processors. Section 8 formalizes the use of usable rules and usable equations with orderings. Section 9 shows the performance of our techniques in practice, after implementing them in the termination tool MU-TERM. Section 10 compares our approach with related work and concludes the paper.

2 Preliminaries

This section collects a number of definitions and notations about term rewriting. More details and missing notions can be found in [4, 22, 27].

Let A be a set and $\mathsf{R} \subseteq \mathsf{A} \times \mathsf{A}$ be a binary relation on A. We denote the transitive closure of R by R^+ and its reflexive and transitive closure by R^*. We say that R is *terminating* (*strongly normalizing*) if there is no infinite sequence $a_1 \mathsf{R} a_2 \mathsf{R} a_3 \cdots$. A reflexive and transitive relation R is a quasi-ordering.

Given relations R and R' over the same set A, we define its *composition* $\mathsf{R} \circ \mathsf{R}'$ as follows: for all $a, b \in \mathsf{A}$, $a \,(\mathsf{R} \circ \mathsf{R}')\, b$ if there is $c \in \mathsf{A}$ such that $a \,\mathsf{R}\, c$ and $c \,\mathsf{R}'\, b$.

Throughout the paper, $\mathcal{X}$ denotes a countable set of variables and Σ (equivalently $\mathcal{F}$ and Γ) denotes a signature, i.e., a set of function symbols $\{f, g, \ldots\}$, each having a fixed arity given by a mapping $ar : \Sigma \to \mathbb{N}$. The set of terms built from Σ and $\mathcal{X}$ is $\mathcal{T}(\Sigma, \mathcal{X})$. $\mathcal{V}ar(t)$ is the set of variables occurring in a term t.

Terms are viewed as labelled trees in the usual way. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. We denote the empty sequence by Λ. Given positions p, q, we denote their concatenation as $p.q$. Positions are ordered by the standard prefix ordering: $p \leq q$ if $\exists q'$ such that $q = p.q'$. If p is a position, and Q is a set of positions, then $p.Q = \{p.q \mid q \in Q\}$. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of nonvariable symbols in t are denoted as $\mathcal{P}os_\Sigma(t)$, and $\mathcal{P}os_\mathcal{X}(t)$ are the positions

of variables. The subterm at position p of t is denoted as $t|_p$, and $t[s]_p$ is the term t with the subterm at position p replaced by s.

We write $s \trianglerighteq t$, read *t is a subterm of s*, if $t = s|_p$ for some $p \in \mathcal{P}os(s)$ and $s \triangleright t$ if $s \trianglerighteq t$ and $s \neq t$. We write $s \not\trianglerighteq t$ and $s \not\triangleright t$ for the negation of the corresponding properties. The symbol labeling the root of t is denoted as $root(t)$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with a 'hole' $\Box$ (a fresh constant symbol). We write $C[\,]_p$ to denote that there is a (usually single) hole $\Box$ at position p of C. Generally, we write $C[\,]$ to denote an arbitrary context and make the position of the hole explicit only if necessary. $C[\,] = \Box$ is called the *empty* context.

A substitution is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\Sigma, \mathcal{X})$. Denote as ε the 'identity' substitution: $\varepsilon(x) = x$ for all $x \in \mathcal{X}$. The set $\mathcal{D}om(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$ is called the *domain* of σ. A *renaming* is an injective substitution ρ such that $\rho(x) \in \mathcal{X}$ for all $x \in \mathcal{X}$. A substitution σ such that $\sigma(s) = \sigma(t)$ for two terms $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$ is called a *unifier* of s and t; we also say that s and t unify (with substitution σ). If two terms s and t unify, then there is a unique *most general unifier* σ (up to renaming of variables) such that for every other unifier τ, there is a substitution θ such that $\theta \circ \sigma = \tau$.

A binary relation $\mathsf{R} \subseteq \mathcal{T}(\Sigma, \mathcal{X}) \times \mathcal{T}(\Sigma, \mathcal{X})$ on terms is *stable* if, for all terms $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ and substitutions σ, we have $\sigma(s)\,\mathsf{R}\,\sigma(t)$ whenever $s\,\mathsf{R}\,t$. We say that the relation is *monotonic* if, for all $f \in \Sigma$, and $s, t, t_1, \ldots, t_k \in \mathcal{T}(\Sigma, \mathcal{X})$, $f(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_k)\,\mathsf{R}\,f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_k)$ whenever $s\,\mathsf{R}\,t$.

A rewrite rule is an ordered pair (l, r), written $l \to r$, with $l, r \in \mathcal{T}(\Sigma, \mathcal{X})$, $l \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$. The left-hand side (*lhs*) of the rule is l, and the right-hand side (*rhs*) is r. A rewrite rule $l \to r$ is said to be *collapsing* if $r \in \mathcal{X}$. A *Term Rewriting System* (TRS) is a pair $\mathcal{R} = (\Sigma, R)$, where R is a set of rewrite rules. An instance $\sigma(l)$ of a *lhs* l of a rule is called a *redex*. Given $\mathcal{R} = (\Sigma, R)$, we consider Σ as the disjoint union $\Sigma = \mathcal{C} \uplus \mathcal{D}$ of symbols $c \in \mathcal{C}$ (called *constructors*) and symbols $f \in \mathcal{D}$ (called *defined functions*), where $\mathcal{D} = \{root(l) \mid l \to r \in R\}$ and $\mathcal{C} = \mathcal{F} - \mathcal{D}$.

A term $s \in \mathcal{T}(\Sigma, \mathcal{X})$ rewrites to t (at position p), written $s \xrightarrow{p}_R t$ (or just $s \to_R t$, or $s \to t$), if $s|_p = \sigma(l)$ and $t = s[\sigma(r)]_p$, for some rule $l \to r \in R$, $p \in \mathcal{P}os(s)$ and substitution σ. We write $s \xrightarrow{>p}_{\mathcal{R}} t$ if $s \xrightarrow{q}_{\mathcal{R}} t$ for some $q > p$. A TRS $\mathcal{R}$ is terminating if its one step rewrite relation $\to_R$ is terminating.

3 Rewriting Modulo Equational Theories

Given a set of equations E, we write $s \overset{p}{\vdash\!\dashv}_E t$ (a single 'equational step') if there is a position $p \in \mathcal{P}os(s)$, an equation $u = v \in E$ and a substitution σ such that $s|_p = \sigma(u)$ and $t|_p = \sigma(v)$, or $s|_p = \sigma(v)$ and $t|_p = \sigma(u)$ (we write $s \vdash\!\dashv_E t$ if position p is not relevant). Note that $\vdash\!\dashv_E$ is a symmetric relation. Then, $\sim_E$ is the reflexive and transitive closure of $\vdash\!\dashv_E$; we have the following equivalence that will be useful in our development:

$$\sim_E = \vdash\!\dashv^*_E = (\overset{\Lambda}{\vdash\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\dashv}_E)^*.$$

We also write $s \overset{>\Lambda}{\sim}_E t$ iff $s = f(s_1, \ldots, s_k)$, $t = f(t_1, \ldots, t_k)$ and $s_i \sim_E t_i$ for all i, $1 \leq i \leq k$. We define $s \overset{\Lambda}{\sim}_E t$ as the reflexive and transitive closure of $\overset{\Lambda}{\vdash}_E$.

Given a rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where R is a set of rewrite rules and E is a set of equational axioms, we write $s \to_{R/E} t$ if there exist u, v such that $s \sim_E u$, $u \to_R v$, and $v \sim_E t$.We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is *E-terminating*, iff $\to_{R/E}$ is terminating. In general, given terms s and t, the problem of checking whether $s \to_{R/E} t$ holds is undecidable: in order to check whether $s \to_{R/E} t$ we have to search through the possibly infinite equivalence classes $[s]_E$ and $[t]_E$ to see whether a matching is found for a subterm of some $u \in [s]_E$ and the result of rewriting u belongs to the equivalence class $[t]_E$. For this reason, a much simpler relation $\to_{R,E}$ is defined, which becomes decidable if an E-matching algorithm exists. For any terms s, t, $s \to_{R,E} t$ holds iff there is a position p in s, a rule $l \to r$ in R, and a substitution σ such that $s|_p \sim_E \sigma(l)$ and $t = s[\sigma(r)]_p$ (see [23]). This relation only allows applying rules from R in redexes at positions equal or above of positions of terms where equations from E have been applied. We say that a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ is *(R, E)-terminating*, if $\to_{R,E}$ is terminating. In the following, we assume that E and R are finite sets of equations and rules, respectively.

Regarding E-termination analysis using *dependency pairs* (DPs), Kusakari and Toyama observed that there is no simple extension of DPs to directly deal with $\to_{R/E}$-computations [18, 16]. In contrast, several approaches have been developed for $\to_{R,E}$-computations [10, 18, 20]. Since $\to_{R,E} \subseteq \to_{R/E}$ (but the opposite inclusion does not hold, in general), E-termination cannot be concluded from (R, E)-termination. Actually, Marché and Urbain showed that there are (R, E)-terminating rewrite theories $\mathcal{R}$ which are *not* E-terminating.

Example 2 *Consider the following rewrite theory $\mathcal{R} = (\Sigma, E, R)$, where '+' is an AC symbol [20]:*

$$a + b \to a + (b + c).$$

Note that $t = a + (b + c)$ is an $\to_{R,E}$-normal form (hence (R, E)-terminating). However, $t \sim_{AC} (a + b) + c$ which is E-nonterminating.

Giesl and Kapur [10] proved the equivalence of both notions of termination with respect to a notion of *extension completion* $\mathcal{E}xt_E(R)$ (see below) of a rewrite theory $\mathcal{R} = (\Sigma, E, R)$ for E *regular* (i.e., $\mathcal{V}ar(u) = \mathcal{V}ar(v)$ for all $u = v$ in E), and *linear* (neither u nor v have repeated variables). For E being a set containing associative or commutative axioms, this notion of extension goes back to Peterson and Stickel [23].

Theorem 1 [10, Theorem 11] *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory with E a regular and linear equational theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, t starts an infinite $\to_{R/E}$-reduction if and only if t starts an infinite $\to_{\mathcal{E}xt_E(R),E}$-reduction. Therefore, $\mathcal{R}$ is E-terminating if and only if $\to_{\mathcal{E}xt_E(R),E}$ is terminating.*

3.1 Combination of Associative and Commutative Theories

Let E be a set of equations that has the modular decomposition $E = \bigcup_{f\in\Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$, where:

- A_f is the associativity axiom $f(f(x, y), z) = f(x, f(y, z))$,
- C_f is the commutativity axiom $f(x, y) = f(y, x)$.

We also define $\Sigma = \Sigma_A \uplus \Sigma_C \uplus \Sigma_{AC} \uplus \Sigma_\varnothing$ where $f \in \Sigma_A \Leftrightarrow E_f = \{A_f\}$, $f \in \Sigma_C \Leftrightarrow E_f = \{C_f\}$, $f \in \Sigma_{AC} \Leftrightarrow E_f = \{A_f, C_f\}$, $f \in \Sigma_\varnothing \Leftrightarrow E_f = \varnothing$. In the following, we often say that a symbol $f \in \Sigma$ *is associative* iff $f \in \Sigma_A \cup \Sigma_{AC}$.

Definition 1 ($A\vee C$-rewrite theory) *An equational theory $E = \bigcup_{f\in\Sigma} E_f$, where if $k = ar(f) \neq 2$, then $E_f = \varnothing$, and if $k = 2$, then $E_f \subseteq \{A_f, C_f\}$ is called an* $A\vee C$-theory. *A rewrite theory $\mathcal{R} = (\Sigma, E, R)$ such that E is an $A\vee C$-theory, is called an* $A\vee C$-rewrite theory.

To deal with rewriting modulo $A\vee C$-theories by using (R, E)-rewriting we have to extend R by following [23, Definition 10.4]:

$$\begin{array}{rcl} \mathcal{E}xt_{AC}(R) & = & R \cup \{f(l, w) \to f(r, w) \mid l \to r \in R, f = root(l) \in \Sigma_{AC}\} \\ \mathcal{E}xt_{A}(R) & = & R \cup \{f(l, w) \to f(r, w), f(w, l) \to f(w, r), f(z, f(l, w)) \to f(z, f(r, w)) \\ & & \mid l \to r \in R, f = root(l) \in \Sigma_A\} \\ \mathcal{E}xt_{C}(R) & = & R \end{array}$$

where w and z are fresh variables which do not occur in the original rule of R. Therefore, given an $A\vee C$ theory E, we let:

$$\mathcal{E}xt_E(R) = \mathcal{E}xt_{AC}(R) \cup \mathcal{E}xt_A(R) \cup \mathcal{E}xt_C(R).$$

Note that $R \subseteq \mathcal{E}xt_E(R)$.

Example 3 *Consider the following TRS $\mathcal{R}$:*

$$f(x, x) \quad \to \quad f(0, 0)$$

where $f \in \Sigma_{AC}$.

Hence, $\mathcal{E}xt_{AC}(R)$ only adds the following rule to $\mathcal{R}$:

$$f(f(x, x), y) \quad \to \quad f(f(0, 0), y)$$

3.2 Minimal Terms and Infinite Rewrite Sequences

Given a TRS $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$, with $\mathcal{C}$ a subsignature of constructors and $\mathcal{D}$ a subsignature of defined symbols, so that each rule in R is of the form $f(t_1, \ldots, t_n) \to r$ with $f \in \mathcal{D}$, the *minimal* nonterminating terms associated to $\mathcal{R}$ are those nonterminating terms t whose proper subterms u (i.e., $t \rhd u$) are terminating. Let $\mathcal{T}_\infty$ denote the set of minimal nonterminating terms associated to $\mathcal{R}$ [14]. Minimal nonterminating terms have two important properties:

1. Every nonterminating term s contains a minimal nonterminating subterm $t \in \mathcal{T}_\infty$ (i.e., $s \trianglerighteq t$), and

2. minimal nonterminating terms t are always rooted by a *defined* symbol $f \in \mathcal{D}$: $\forall t \in \mathcal{T}_\infty, root(t) \in \mathcal{D}$.

Considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t = f(t_1, \ldots, t_k) \in \mathcal{T}_\infty$ is helpful to arrive at the notion of dependency pair. Such sequences proceed as follows (see, e.g., [14]):

1. a finite number of reductions can be performed *below* the root of t, thus rewriting t into t'; then

2. a rule $f(l_1, \ldots, l_k) \to r$ applies *at the root* of t' (i.e., $t' = \sigma(f(l_1, \ldots, l_k))$ for some substitution σ); and

3. there is a minimal nonterminating term $u \in \mathcal{T}_\infty$ (hence $root(u) \in \mathcal{D}$) at some position p of $\sigma(r)$ which is a *nonvariable position of* r which 'continues' the infinite sequence initiated by t in a similar way.

This means that considering the occurrences of defined symbols in the right-hand sides of the rewrite rules suffices to 'catch' *every possible infinite rewrite sequence starting from* $\sigma(r)$. In particular, no infinite sequence can be issued from t' *below* the variables of r (more precisely: all bindings $\sigma(x)$ are *terminating* terms). The standard definition of dependency pair [1] and (minimal) chain of dependency pairs [13] rely on (1)–(3) above [14]. These facts are formalized as follows:

Proposition 1 [14, Lemma 1] *Let $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS. For all $t \in \mathcal{T}_\infty$, there exist $l \to r \in R$, a substitution σ and a term $u \in \mathcal{T}_\infty$ such that $root(u) \in \mathcal{D}$, $t \xrightarrow{>\Lambda}{}^* \sigma(l) \xrightarrow{\Lambda} \sigma(r) \trianglerighteq u$ and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.*

The following auxiliary results will be used later.

Proposition 2 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory and $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If t is (R, E)-terminating, then*

1. *If $t \trianglerighteq s$, then s is (R, E)-terminating.*

2. *If $t \to_{R,E}^* s$ then s is (R, E)-terminating.*

Proof. Trivial. □

Proposition 3 (E-Termination Preserved under E-Equivalence) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory and $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If $t \sim_E s$, then t is E-terminating if and only if s is E-terminating.*

PROOF. Trivial. □

Proposition 3 does *not* hold if we change E-termination by (R, E)-termination (see Example 2). However, as a consequence of Theorem 1 and Proposition 3, we have:

Corollary 1 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory such that E is a set of regular and linear equations and $t, s \in \mathcal{T}(\Sigma, \mathcal{X})$. If $t \sim_E s$, then t is $(\mathcal{E}xt_E(R), E)$-terminating if and only if s is $(\mathcal{E}xt_E(R), E)$-terminating.*

As a corollary of Theorem 1, we have the following.

Corollary 2 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, t is E-terminating if and only if it is $(\mathcal{E}xt_E(R), E)$-terminating.*

In the following section we begin the analysis of infinite E-rewrite sequences according to the schema in [14]. We aim at providing an appropriate notion of minimal E-nonterminating term (for $A \vee C$-theories E) which allows us to reach a result similar to Proposition 1.

4 Stably Minimal E-nonterminating Terms

In the dependency pair approach [1, 14, 13], the analysis of infinite rewrite sequences is restricted to those starting from *minimal nonterminating terms* $t \in \mathcal{T}_\infty$. The following notion of minimal E-nonterminating term is implicit in [10, proof of Theorem 16]. Similar definitions can be found in [17, 18, 16, 21].

Definition 2 (Minimal E-nonterminating Term [10]**)** *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. An E-nonterminating term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ is said to be* minimal *(written $t \in \mathcal{T}_{\infty,R,E}$) if every strict subterm s of t (i.e., $t \rhd s$) is $(\mathcal{E}xt_E(R), E)$-terminating.*

Remark 1 *In Definition 2, if we assume that E is linear and regular (like $A \vee C$-theories), then, by Theorem 1, we could equivalently start by saying that t is $(\mathcal{E}xt_E(R), E)$-nonterminating. This leads to a more symmetric definition, which we often use in the following without further comment.*

Every E-nonterminating term s contains a minimal E-nonterminating subterm $t \in \mathcal{T}_{\infty,R,E}$ (this is stated without proof in [10, proof of Theorem 16]).

Proposition 4 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory and $s \in \mathcal{T}(\Sigma, \mathcal{X})$. If s is E-nonterminating, then there is a subterm t of s ($s \trianglerighteq t$) such that $t \in \mathcal{T}_{\infty,R,E}$.*

PROOF. By structural induction. If s is a constant symbol, it is obvious: take $t = s$. If $s = f(s_1, \ldots, s_k)$, then we proceed by contradiction. If there is no (not necessarily strict) subterm t of s such that t is minimal, then in particular s is not minimal. Therefore, there is a strict subterm t of s ($s \rhd t$) which is

E-nonterminating. By the Induction Hypothesis, there is t' which is minimal and such that $t \trianglerighteq t'$. Then, we have $s \triangleright t'$, thus leading to a contradiction. □

Note that Giesl and Kapur's minimality of terms is preserved under $\rightarrow_{\mathcal{E}xt_E(R),E}$-reductions below the root.

Proposition 5 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory and $s \in \mathcal{T}_{\infty,R,E}$. If $s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} t$ and t is E-nonterminating, then $t \in \mathcal{T}_{\infty,R,E}$.*

PROOF. Since s is rewritten below the root, we can write $s = f(s_1, \ldots, s_k)$, where, by minimality of s, we know that $s_1, \ldots, s_k$ are all $(\mathcal{E}xt_E(R), E)$-terminating. Furthermore, since $\xrightarrow{>\Lambda}_{\mathcal{E}xt_E(R),E}$ performs no rewriting or $\sim_E$-steps at the root, we have that $t = f(t_1, \ldots, t_k)$ with $s_i \rightarrow^*_{\mathcal{E}xt_E(R),E} t_i$ for all i, $1 \leq i \leq k$. By Proposition 2, t_i is $(\mathcal{E}xt_E(R), E)$-terminating for all i, $1 \leq i \leq k$. And since t is assumed to be E-nonterminating, $t \in \mathcal{T}_{\infty,R,E}$. □

Remark 2 (Root Symbols of Minimal Terms) *Note that if E is an $A \vee C$-equational theory, then $root(t) \in \mathcal{D}$ whenever $t \in \mathcal{T}_{\infty,R,E}$. As remarked by Giesl and Kapur (see also Example 8 below) this is not true for arbitrary equational theories.*

The problem with Giesl and Kapur's Definition 2 is that minimality is *not* preserved under E-equivalence.

Example 4 *Consider again the TRS $\mathcal{R}$ in Example 3.*
Following [10], the term $f(f(1,0),0) \in \mathcal{T}_{\infty,R,E}$ since is AC-nonterminating

$$f(f(1,0),0) \sim_{AC} f(1,f(0,0)) \sim_{AC} f(f(0,0),1) \xrightarrow{\Lambda}_R f(f(0,0),1) \cdots$$

but its strict subterms $f(1,0)$, 1 and 0 are AC-terminating. However, the root step with $\sigma(l) = \sigma(f(f(x,x),y) = f(f(0,0),1)$ shows that $\sigma(l) \notin \mathcal{T}_{\infty,R,E}$ since $f(0,0)$ is AC-nonterminating.

Example 5 *Consider the following TRS $\mathcal{R}$:*

$$f(x,x) \quad \rightarrow \quad f(0,f(1,2)) \tag{1}$$

where $f \in \Sigma_{AC}$. Hence, $\mathcal{E}xt_{AC}(R)$ only adds the following rule to $\mathcal{R}$:

$$f(f(x,x),y) \quad \rightarrow \quad f(f(0,f(1,2)),y) \tag{2}$$

Note that $t = f(f(0,1),f(0,f(1,2)))$ is $(\mathcal{E}xt_{AC}(R), AC)$-nonterminating:

$$\begin{array}{rll}
\underline{f(f(0,1),f(0,f(1,2)))} & \sim_A & f(0,\underline{f(1,f(0,f(1,2)))}) \\
& \sim_A & f(0,f(\underline{f(1,0)},f(1,2))) \\
& \sim_C & f(0,f(\underline{f(0,1)},f(1,2))) \\
& \sim_A & \underline{f(0,f(0,f(1,f(1,2))))} \\
& \sim_A & \underline{f(f(0,0),f(1,f(1,2)))} \\
& \xrightarrow{\Lambda}_{\mathcal{E}xt_{AC}(R)} & \underline{f(f(0,f(1,2)),f(1,f(1,2)))} \\
& \rightarrow_{\mathcal{E}xt_{AC}(R),AC} & \cdots
\end{array}$$

Since $f(0,1)$ *and* $f(0,f(1,2))$ *are in* $(\mathcal{E}xt_{AC}(R),AC)$*-normal form, we have that* $t \in \mathcal{T}_{\infty,R,AC}$*. However,* $t' = f(f(0,0),f(1,f(1,2)))$*, which is* AC*-equivalent to* t *(i.e.,* $t \sim_{AC} t'$*), is* AC*-nonterminating, but it is* not *minimal because its strict subterm* $f(1,f(1,2)))$ *is* $(\mathcal{E}xt_{AC}(R),AC)$*-nonterminating:*

$$
\begin{array}{rcl}
\underline{f(1,f(1,2))} & \sim_A & \underline{f(f(1,1),2)} \\
& \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_{AC}(R)} & \underline{f(f(0,f(1,2)),2)} \\
& \sim_A & \underline{f(0,f(f(1,2),2))} \\
& \sim_A & \underline{f(0,f(1,f(2,2)))} \\
& \sim_A & \underline{f(f(0,1),f(2,2))} \\
& \sim_C & \underline{f(f(2,2),f(0,1))} \\
& \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_{AC}(R)} & f(\underline{f(0,f(1,2))},f(0,1)) \\
& \sim_A & \underline{f(f(f(0,1),2)),f(0,1))} \\
& \sim_C & \underline{f(f(0,1),f(f(0,1),2)))} \\
& \sim_A & \underline{f(f(0,1),f(0,f(1,2)))} \\
& \rightarrow_{\mathcal{E}xt_{AC}(R),AC} & \cdots
\end{array}
$$

Example 5 shows that an essential property of minimal terms when considered as part of infinite $(\mathcal{E}xt_E(R),E)$-rewriting sequences for $A\vee C$-theories E gets lost: the application of $(\mathcal{E}xt_E(R),E)$-rewrite steps *at the root* of a minimal term s by means of a rule $l \rightarrow r$ (i.e., $s \sim_{AC} \sigma(l) \stackrel{\Lambda}{\rightarrow}_{\mathcal{E}xt_E(R)} \sigma(r)$) does *not* guarantee that there is a *nonvariable subterm* v of the right-hand side r which is a prefix of the 'next' minimal term in the infinite sequence. In the following proposition, we prove that the problem illustrated in Example 5 is due to the application of associative steps at the root of a minimal term.

Proposition 6 *Let* $\mathcal{R} = (\Sigma,E,R)$ *be a rewrite theory and* $t \in \mathcal{T}_{\infty,R,E}$*.*

1. *If* E *is regular and linear and* $t' \stackrel{>\Lambda}{\sim}_E t$*, then* $t' \in \mathcal{T}_{\infty,R,E}$*.*
2. *If* $C_{root(t)} \in E_{root(t)}$ *and* $t' \stackrel{\Lambda}{\sim}_C t$*, then* $t' \in \mathcal{T}_{\infty,R,E}$*.*

Proof. Let $t = f(t_1,\ldots,t_k)$. By minimality of t, t_i is $(\mathcal{E}xt_E(R),E)$-terminating for all $1 \leq i \leq k$.

1. If $t' \stackrel{>\Lambda}{\sim}_E t$, then $t' = f(t'_1,\ldots,t'_k)$ and $t'_i \sim_E t_i$ for all $1 \leq i \leq k$. By Proposition 3, t' is $(\mathcal{E}xt_E(R),E)$-nonterminating and by Corollary 1, t'_i is $(\mathcal{E}xt_E(R),E)$-terminating for all $1 \leq i \leq k$. Hence $t' \in \mathcal{T}_{\infty,R,E}$.
2. If $C_{root(t)} \in E_{root(t)}$, then $k = 2$ and $t = f(t_1,t_2)$, where both t_1 and t_2 are $(\mathcal{E}xt_E(R),E)$-terminating. Since $t' = f(t_2,t_1) \stackrel{\Lambda}{\sim}_C t$, by Proposition 3 t' is $(\mathcal{E}xt_E(R),E)$-nonterminating and we have $t' \in \mathcal{T}_{\infty,R,E}$.

□

Example 6 *Term t in Example 5 can be rewritten at the root* only *by rule* (2) *of $\mathcal{E}xt_{AC}(R)$. We can apply this rule to t' in Example 5 (for instance) to obtain $s' = \sigma(r) = f(f(0, f(1,2)), f(1, f(1,2)))$ (where $r = f(f(0, f(1,2)), y)$), which is $(\mathcal{E}xt_{AC}(R), AC)$-nonterminating. Note that s' contains a minimal term $u \in \mathcal{T}_{\infty,R,E}$. Since $s'|_2 = f(1, f(1,2))$ is $(\mathcal{E}xt_{AC}(R), AC)$-nonterminating, it follows that s' is* not *minimal. Since $s'|_1 = f(0, f(1,2))$ is $(\mathcal{E}xt_{AC}(R), AC)$-terminating, the only possibility is that u occurs in $s'|_2$. Actually, $s'|_2$ is minimal already; hence, $u = s'|_2$. But note the absence of any nonvariable position $p \in \mathcal{P}os(r)$ in the right-hand side of the considered rule such that $\sigma(r|_p) = u = f(1, f(1,2))$.*

This is in sharp contrast with the situation of the DP-approach for ordinary rewriting. Furthermore, it is not difficult to see that for all $t'' \sim_{AC} t$ such $t'' = \sigma'(l)$ for some substitution σ', we have a similar situation. Thus, the problem illustrated here cannot be solved by using a different $\sim_{AC}$ sequence before performing the $\mathcal{E}xt_{AC}(R)$-root-step.

In the following we introduce a new notion of minimality which solves these problems.

4.1 A New Notion of Minimal E-nonterminating Terms

The following definition solves the problems discussed above by explicitly requiring that the condition defining minimality is preserved under E-equivalence.

Definition 3 (Stably Minimal E-nonterminating Term) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. Let $\mathcal{M}_{\infty,R,E}$ be a set of stably minimal E-nonterminating terms in the following sense: $t \in \mathcal{T}(\Sigma, \mathcal{X})$ belongs to $\mathcal{M}_{\infty,R,E}$ iff t is E-nonterminating, and for all $t' \sim_E t$ and every proper subterm s' of t' (i.e., $t' \rhd s'$), s' is $(\mathcal{E}xt_E(R), E)$-terminating.*

We have the following useful characterization of minimality.

Proposition 7 (Characterization of Stably Minimal Terms) *Let $\mathcal{R} = (\Sigma, R, E)$ be a rewrite theory and $t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, $t \in \mathcal{M}_{\infty,R,E}$ if and only if $[t]_E \subseteq \mathcal{T}_{\infty,R,E}$. Therefore,*

$$\mathcal{M}_{\infty,R,E} = \{t \in \mathcal{T}(\Sigma, \mathcal{X}) \mid [t]_E \subseteq \mathcal{T}_{\infty,R,E}\}$$

The problem in Example 5 disappears now: t is *not* (stably) minimal according to Definition 3. The same situation happens with the problem in Example 4: $f(f(1,0),0) \in \mathcal{T}_{\infty,R,E}$ but $f(f(1,0),0) \notin \mathcal{M}_{\infty,R,E}$ since $f(f(1,0),0) \sim_E f(f(0,0),1)$ and $f(0,0)$ is E-nonterminating. In fact, $f(0,0) \in \mathcal{M}_{\infty,R,E}$.

The following result shows how to *find* stably minimal E-nonterminating terms associated to a given E-nonterminating term. This is essential in our development. A set of equations E is *size-preserving* if and only if for each equation $u = v$ the length of u and v are the same, i.e. $|u| = |v|$ and the multiset of the variables in u coincides with the multiset of the variables in v [22].

Proposition 8 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory such that E is regular and size-preserving. Let $s \in \mathcal{T}(\Sigma, \mathcal{X})$. If s is E-nonterminating, then there is a subterm t of some $s' \sim_E s$ (i.e., $s' \trianglerighteq t$) such that $t \in \mathcal{M}_{\infty,R,E}$.*

Proof. By structural induction. If s is a constant symbol or a variable, then since s has no strict subterms, then $s \in \mathcal{M}_{\infty,R,E}$, so in this case, we can choose $t = s$. If $s = f(s_1, \ldots, s_k)$, then we proceed by contradiction. If there is no subterm t of some $s' \sim_E s$ ($s' \trianglerighteq t$) such that $t \in \mathcal{M}_{\infty,R,E}$, then in particular $s \notin \mathcal{M}_{\infty,R,E}$, (and thus $s' \notin \mathcal{M}_{\infty,R,E}$ for all $s' \sim_E s$) i.e., (since s is E-nonterminating) there is an E-equivalent term $s' \sim s$ containing a strict $(\mathcal{E}xt_E(R), E)$-nonterminating subterm t' ($s' \triangleright t'$). Therefore, t' is E-nonterminating as well. By the Induction Hypothesis, there is $t \in \mathcal{M}_{\infty,R,E}$ such that $t' \trianglerighteq t$. Then, $s' \triangleright t$, thus leading to a contradiction. □

Clearly, Proposition 8 holds whenever $\mathcal{R}$ is an $A \vee C$-rewrite theory.

Example 7 *Consider the term t in Example 5. Although $t \in \mathcal{T}_{\infty,R,E}$, $t \notin \mathcal{M}_{\infty,R,E}$: the term $t' = f(f(0,0), f(1, f(1,2)))$, which is AC-equivalent to t, contains a subterm $u = f(1, f(1,2))$ which is E-nonterminating. It is not difficult to see that actually $u \in \mathcal{M}_{\infty,R,E}$.*

In general, Proposition 8 does *not* hold for arbitrary sets of equations E.

Example 8 *Consider the following example [10, Example 13]:*

$$R : f(x) \quad \to \quad x \qquad\qquad E : f(a) \quad = \quad a$$

Note that $a \in \mathcal{T}_{\infty,R,E}$. However, a is not *stably minimal because $a \sim_E f(a)$ but $f(a) \notin \mathcal{T}_{\infty,R,E}$. Thus, Proposition 8 does not hold.*

Since $\mathcal{M}_{\infty,R,E} \subseteq \mathcal{T}_{\infty,R,E}$, for $A \vee C$-rewrite theories E we have the following corollary of Proposition 5.

Corollary 3 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory and $s \in \mathcal{M}_{\infty,R,E}$. If $s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} t$ and t is E-nonterminating, then $t \in \mathcal{T}_{\infty,R,E}$.*

In general, Corollary 3 does *not* hold if we require that $t \in \mathcal{M}_{\infty,R,E}$.

Example 9 *Term $u = f(f(1,1),2)$ in Example 6 is stably minimal: $u \in \mathcal{M}_{\infty,R,E}$. We have that $f(\underline{f(1,1)},2) \xrightarrow{>\Lambda}_R f(f(0, f(1,2)),2)$. Note that $f(f(0, f(1,2)),2) \notin \mathcal{M}_{\infty,R,E}$. We have*

$$\underline{f(f(0, f(1,2)),2)} \sim_A f(0, \underline{f(f(1,2),2)}) \sim_A f(0, f(1, f(2,2)))$$

where $f(0, f(1, f(2,2)))$ contains a subterm $f(1, f(2,2))$ which is $(\mathcal{E}xt_E(R), E)$-nonterminating.

The following results show that the problem arises when $s \in \mathcal{M}_{\infty,R,E}$ is such that $root(s)$ includes *associativity* among its axioms, that is, $A_f \in E_f$. In the following, we focus on $A \vee C$-rewrite theories. Hence, we will often implicitly use Corollary 2 to speak about E-termination rather than $(\mathcal{E}xt_E(R), E)$-termination (see Remark 1).

Proposition 9 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. If $t \in \mathcal{T}_{\infty,R,E}$ is such that (1) $A_{root(t)} \notin E_{root(t)}$ or (2) $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then $t \in \mathcal{M}_{\infty,R,E}$.*

PROOF. Let $t = f(t_1, \ldots, t_k)$, where, since $t \in \mathcal{T}_{\infty,R,E}$, t_i is E-terminating for all i, $1 \leq i \leq k$. We consider two main cases according to f:

1. If f does not have the associativity axiom, i.e., $A_f \notin E_f$, then we consider two cases:
   (a) f is commutative, i.e., $E_f = \{C_f\}$. Then, $k = 2$ and we can write $t = f(t_1, t_2)$, where t_1 and t_2 are E-terminating. For all $u \sim_E t$ given by $u = f(u_1, u_2)$ we have two possibilities: either $u_1 \sim_E t_1$ and $u_2 \sim_E t_2$, or $u_1 \sim_E t_2$ and $u_2 \sim_E t_1$. In both cases, since E-equivalence preserves E-termination (Proposition 3), we conclude that u_1 and u_2 are E-terminating and hence $t \in \mathcal{M}_{\infty,R,E}$.
   (b) f is not commutative, i.e., $E_f = \varnothing$. Then, for all $u \sim_E t$ we have $u \overset{>\Lambda}{\sim}_E t$ and the result follows from Proposition 6-(1).
2. If $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then no associativity axiom can be applied at the root of t. Then, we can treat t as in one of the cases 1a or 1b above.

□

Proposition 10 *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A \vee C$-rewrite theory and $s \in \mathcal{M}_{\infty,R,E}$ be such that (1) $A_{root(s)} \notin E_{root(s)}$ or (2) $s = f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1), root(s_2) \in \mathcal{C}$. If $s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_{E(R),E}} t$ and t is E-nonterminating, then $t \in \mathcal{M}_{\infty,R,E}$.*

PROOF. Since $s \in \mathcal{M}_{\infty,R,E}$, we have that, for all $s' \sim_E s$, all proper subterms u of s' are E-terminating. We can write $s = f(s_1, \ldots, s_k)$, where, by stable minimality of s, all the $s_1, \ldots, s_k$ are E-terminating. Furthermore, since $\xrightarrow{>\Lambda}_{\mathcal{E}xt_{E(R),E}}$ performs no rewriting or E-steps at the root, we have that $t = f(t_1, \ldots, t_k)$ with $s_i \rightarrow^*_{\mathcal{E}xt_{E(R),E}} t_i$ for all i, $1 \leq i \leq k$. By Proposition 2, t_i is E-terminating for all i, $1 \leq i \leq k$. Therefore, $t \in \mathcal{T}_{\infty,R,E}$. If $root(s) = root(t)$ is such that $A_f \notin E_f$, by Proposition 9, $t \in \mathcal{M}_{\infty,R,E}$. On the other hand, if $s = f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1), root(s_2) \in \mathcal{C}$, then reductions on s_1 and s_2 do not change $root(s_1)$ nor $root(s_2)$. Thus, $t = f(t_1, t_2)$ and $root(t_1) = root(s_1)$ and $root(t_2) = root(s_2)$. Since $f \in \mathcal{D}$ (due to minimality of s), by Proposition 9, $t \in \mathcal{M}_{\infty,R,E}$. □

Now we provide a more precise result about where we can find stably minimal subterms within an E-nonterminating term for $A \vee C$-rewrite theories $\mathcal{R} = (\Sigma, E, R)$. In the following theorem, given a term s and a symbol f, by an *f-subterm t of s* (written $s \trianglerighteq_f t$) we mean a subterm t of s such that $t = s|_p$ and for all $q < p$, $root(s|_q) = f$. We also write $s \triangleright_f t$ if $s \trianglerighteq_f t$ and $s \neq t$. This notion

is similar to the one used in [18] called *head subterm* but taking into account s instead of $[s]_E$ to get the subterm.

Theorem 2 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. If s is E-nonterminating, then there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s ($s \trianglerighteq t$) and*

1. *If (1) $A_{root(t)} \notin E_{root(t)}$ or (2) $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then $t \in \mathcal{M}_{\infty,R,E}$.*
2. *If $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and $t \notin \mathcal{M}_{\infty,R,E}$, then there is $s' \sim_E t$ and a strict f-subterm u of s' (i.e., $s' \rhd_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

Proof. By Proposition 4, s contains a subterm $t \in \mathcal{T}_{\infty,R,E}$. If (1) $A_{root(t)} \notin E_{root(t)}$ or (2) $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then, by Proposition 9, $t \in \mathcal{M}_{\infty,R,E}$. Otherwise, we know that $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$. If $t \in \mathcal{M}_{\infty,R,E}$, then we are done. If $t \notin \mathcal{M}_{\infty,R,E}$, then there must be a term $t' \sim_E t$, $t' \neq t$, which contains a strict subterm t'' (i.e., $t' \rhd t''$) which is E-nonterminating. By Proposition 8, there are terms $t''' \sim_E t''$ and $u \in \mathcal{M}_{\infty,R,E}$ such that $t''' \trianglerighteq u$. If $root(u) \neq f$, then, since $t \sim_E t' \rhd t'' \sim_E t''' \trianglerighteq u$, there must be a strict subterm v of t (i.e., $t \rhd v$) satisfying $u \sim_E v$. By Proposition 3, v is E-nonterminating. This contradicts that $t \in \mathcal{T}_{\infty,R,E}$. Thus, $root(u) = f$ as desired. Furthermore, we note that $t' = C[t'']$ for some *nonempty* context C and hence $t \sim_E t' \sim_E C[t''']$. Thus, if we let $s' = C[t''']$, then $s' \rhd u$. We can further conclude that $s' \rhd_f u$: first note that $root(s') = f$ because $s' \sim_E t$, $root(t) = f$, and $\sim_E$-steps do not change the root of t (because E is an $A\vee C$-theory). Assume that s'' is such that $s' \rhd s'' \rhd u$ and $root(s'') \neq f$. Then by reasoning as above, we would conclude that t contains a subterm $v'' \sim_E s''$, which contradicts $t \in \mathcal{T}_{\infty,R,E}$. Thus, $s' \rhd_f u$. This completes the proof. □

The following result is just a convenient reformulation of the previous one.

Corollary 4 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. If s is E-nonterminating, then either there is a subterm $t \in \mathcal{M}_{\infty,R,E}$ of s ($s \trianglerighteq t$), or there is a subterm $t \in \mathcal{T}_{\infty,R,E}$ of s satisfying that $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and such that there is $s' \sim_E t$ and a strict f-subterm u of s' ($s' \rhd_f u$) such that $root(u) = f$ and $u \in \mathcal{M}_{\infty,R,E}$.*

5 Structure of (Stably) Minimal Infinite $A\vee C$-Rewrite Sequences

Now we analyze $A\vee C$-rewrite sequences starting from stably minimal $A\vee C$-nonterminating terms. First we consider a restricted case.

Proposition 11 *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A\vee C$-rewrite theory. Let $s \in \mathcal{M}_{\infty,R,E}$ be such that $f = root(s)$ and either (1) $A_f \notin E_f$, or (2) $s =$*

$f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1)$, $root(s_2) \in \mathcal{C}$. Assume that for all $l \to r \in R$ such that $root(l) = f$ and all subterms v of r ($r \trianglerighteq v$) such that $v = g(v_1, v_2)$ for some associative symbol g, we have that $root(v_1), root(v_2) \notin \mathcal{X} \cup \{g\}$. Then, there exist $l \to r \in R$, a substitution σ and terms $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and $u \in \mathcal{M}_{\infty,R,E}$ such that

$$s \xrightarrow{>\Lambda}{}^{*}_{\mathcal{E}xt_E(R),E} t \sim_E \sigma(l) \xrightarrow{\Lambda}_R \sigma(r) \trianglerighteq u$$

and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$.

PROOF. Let S be an infinite $(\mathcal{E}xt_E(R), E)$-rewrite sequence starting from s. Since $s \in \mathcal{M}_{\infty,R,E}$, s is $(\mathcal{E}xt_E(R), E)$-nonterminating and all its proper subterms are $(\mathcal{E}xt_E(R), E)$-terminating, S must contain a possibly empty sequence of *inner* $(\mathcal{E}xt_E(R), E)$-rewrite steps followed by a root step. Therefore, there exists a rule $l \to r \in \mathcal{E}xt_E(R)$ such that $s \xrightarrow{>\Lambda}{}^{*}_{\mathcal{E}xt_E(R),E} t \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$. By Proposition 10, we know that $t \in \mathcal{M}_{\infty,R,E}$. Furthermore, due to our assumptions (1) or (2) on s, and taking into account the shape of rules in $\mathcal{E}xt_E(R) - R$ for $A\vee C$-theories E, we can conclude that the rule $l \to r$ actually belongs to R. Since stable minimality is preserved under $\sim_E$, we also have $\sigma(l) \in \mathcal{M}_{\infty,R,E}$. Since $\sim_E$-steps do not change the root symbol of terms for $A\vee C$-theories E, $root(s) = root(t) = root(l) \in \mathcal{D}$. Let $l = f(l_1, \ldots, l_k)$ for some k-ary defined symbol $f \in \mathcal{D}$. Since $\sigma(l) \in \mathcal{M}_{\infty,R,E}$, $\sigma(l_i)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all i, $1 \le i \le k$. In particular, $\sigma(x)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all $x \in \mathcal{V}ar(l)$. Since $\sigma(r)$ is $(\mathcal{E}xt_E(R), E)$-nonterminating, by Theorem 2 there is a subterm $u \in \mathcal{T}_{\infty,R,E}$ of $\sigma(r)$. Therefore, there must be a nonvariable subterm v of r (i.e., $r \trianglerighteq v$ and $root(v) \in \mathcal{D}$), such that $u = \sigma(v)$. Let $g = root(v)$. We consider two cases according to Theorem 2:

1. If $A_g \notin E_g$, then $u \in \mathcal{M}_{\infty,R,E}$.
2. If $A_g \in E_g$, then there must be $v = g(v_1, v_2)$ for terms v_1 and v_2 such that $root(v_1), root(v_2) \notin \mathcal{X} \cup \{g\}$. Therefore, $u = g(u_1, u_2)$ with $u_i = \sigma(v_i)$ satisfying $root(u_i) \neq g$ for $i = 1, 2$. Then, $u \in \mathcal{M}_{\infty,R,E}$.

□

Unfortunately, stable minimality of (arbitrary) E-nonterminating terms s for $A\vee C$-theories E is not preserved under inner $(\mathcal{E}xt_E(R), E)$-rewritings (see Example 9). As Proposition 10 shows, the problem arises when s is rewritten into a term like, e.g., $t = f(f(t_1, t_2), t_3)$ on which associative steps can be issued to rearrange t and possibly introducing an E-nonterminating term below the root, thus *losing* stable minimality.

However, as a consequence of previous results, the following theorem establishes the desired property for stable minimal $A\vee C$-nonterminating terms.

Theorem 3 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. For all $s \in \mathcal{M}_{\infty,R,E}$, there exist $l \to r \in \mathcal{E}xt_E(R)$ and a substitution σ such that*

$$s \xrightarrow{>\Lambda}{}^{*}_{\mathcal{E}xt_E(R),E} t \sim_E t' \trianglerighteq_f t'' \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$$

$t'' \in \mathcal{M}_{\infty,R,E}$ and there is a nonvariable subterm v of r ($r \trianglerighteq v$), such that either

1. $v = f(v_1, v_2)$ *for some associative symbol* f, $root(v_1) \in \mathcal{X} \cup \{f\}$ *or* $root(v_2) \in \mathcal{X} \cup \{f\}$, $root(\sigma(v_1)) = f$ *or* $root(\sigma(v_2)) = f$, $\sigma(v) \in \mathcal{T}_{\infty,R,E}$ *and there is a term* $t' \sim_E \sigma(v)$ *containing a strict* f*-subterm* $u = f(u_1, u_2)$ $(t' \rhd_f u)$ *such that* $u \in \mathcal{M}_{\infty,R,E}$, *or*

2. $\sigma(v) \in \mathcal{M}_{\infty,R,E}$ *otherwise.*

PROOF. Let S be an infinite $(\mathcal{E}xt_E(R), E)$-rewrite sequence starting from s. Since $s \in \mathcal{M}_{\infty,R,E}$, s is $(\mathcal{E}xt_E(R), E)$-nonterminating and all its proper subterms are $(\mathcal{E}xt_E(R), E)$-terminating, S must contain a root step after possibly many $(\mathcal{E}xt_E(R), E)$-rewrite steps below the root.

Therefore, $s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} t$ and by Corollary 3 , $t \in \mathcal{T}_{\infty,R,E}$. By Theorem 2, we have that,

1. If $A_{root(t)} \notin E_{root(t)}$ or $t = f(t_1, t_2)$, $A_f \in E_f$, $root(t_1) \neq f$, and $root(t_2) \neq f$, then $t \in \mathcal{M}_{\infty,R,E}$.

2. If $t = f(t_1, t_2)$, $A_f \in E_f$, and $root(t_1) = f$ or $root(t_2) = f$, and $t \notin \mathcal{M}_{\infty,R,E}$, then there is $t' \sim_E t$ and a strict f-subterm t'' of t' (i.e., $t' \rhd_f t''$) such that $root(t'') = f$ and $t'' \in \mathcal{M}_{\infty,R,E}$.

Therefore, the sequence can proceed as follows:

$$t \sim_E t' \trianglerighteq_f t'' \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$$

Where, if (1) holds, $t \in \mathcal{M}_{\infty,R,E}$ and therefore $t = t' = t''$. Otherwise, in case (2), if $t \in \mathcal{M}_{\infty,R,E}$ we are done as before. If not, there is $t' \sim_E t$ and a strict f-subterm t'' of t' (i.e., $t' \rhd_f t''$) such that $root(t'') = f$ and $t'' \in \mathcal{M}_{\infty,R,E}$.

Since stably minimality is preserved by $\sim_E$, therefore, $\sigma(l) \in \mathcal{M}_{\infty,R,E}$. Since $A \vee C$ axioms cannot change $root(s)$ or $root(t)$, we have $f = root(s) = root(t) = root(l) \in \mathcal{D}$. Write $l = f(l_1, \ldots, l_k)$. Since $\sigma(l) \in \mathcal{M}_{\infty,R,E}$, $\sigma(l_i)$ is $(\mathcal{E}xt_E(R), E)$-terminating. In particular, $\sigma(x)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all $x \in \mathcal{V}ar(l)$. Since $\sigma(r)$ is $(\mathcal{E}xt_E(R), E)$-nonterminating, by Proposition 4 there is a subterm $u \in \mathcal{T}_{\infty,R,E}$ of $\sigma(r)$ $(\sigma(r) \trianglerighteq u)$. Since $\sigma(x)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all $x \in \mathcal{V}ar(r)$, there must be a nonvariable subterm v of r $(r \trianglerighteq v)$, such that $u = \sigma(v)$. By Theorem 2, we only need to carefully consider the case when $u = f(u_1, u_2) \notin \mathcal{M}_{\infty,R,E}$ for some associative symbol f such that $A_f \in E_f$, $root(u_1) = f$ or $root(u_2) = f$. Therefore, we must have $v = f(v_1, v_2)$ for some terms v_1 and v_2. Since $u_1 = \sigma(v_1)$ and $u_2 = \sigma(v_2)$, we must have $v_1 \in \mathcal{X} \cup \{f\}$ or $v_2 \in \mathcal{X} \cup \{f\}$. Theorem 2 also ensures that there is $s' \sim_E \sigma(v)$ such that $s' \rhd_f u'$ and $u' \in \mathcal{M}_{\infty,R,E}$. □

Example 5 shows that Theorem 3 does not hold for Giesl and Kapur's minimal terms $s \in \mathcal{T}_{\infty,R,E}$.

6 $A\vee C$-Dependency Pairs and Chains

Propositions 8 and 11 together with Theorem 3 are the basis for our definition of *$A\vee C$-Dependency Pairs* and the corresponding *chains*. Together, they show that given an $A\vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$, every E-nonterminating term s has an associated infinite $(\mathcal{E}xt_E(R), E)$-rewrite sequence starting from a stably minimal subterm $t \in \mathcal{M}_{\infty,R,E}$. Such a sequence proceeds as described in Proposition 11 and Theorem 3, depending on the shape of t.

This process is abstracted in the following definition of $A\vee C$-dependency pairs (Definition 4) and in the definition of chain below (Definition 5).

Given a signature Σ and $f \in \Sigma$, we let $f^\sharp$ denote a fresh new symbol (often called *tuple* symbol or DP-symbol) associated to a symbol f [1]. Let $\Sigma^\sharp$ be the set of tuple symbols associated to symbols in Σ. As usual, for $t = f(t_1, \ldots, t_k) \in \mathcal{T}(\Sigma, \mathcal{X})$, we write $t^\sharp$ to denote the *marked* term $f^\sharp(t_1, \ldots, t_k)$ (written sometimes $F(t_1, \ldots, t_k)$). Given a set of rules R and a symbol $f \in \Sigma$, we let $R_f = \{l \to r \in R \mid root(l) = f\}$.

Definition 4 ($A\vee C$-Dependency Pairs) *Let $\mathcal{R} = (\Sigma, E, R) = (\mathcal{C} \uplus \mathcal{D}, E, R)$ be an $A\vee C$-rewrite theory. Then, $\mathsf{DP}_E(R) = \{l^\sharp \to s^\sharp \mid l \to r \in \mathcal{E}xt_E(R), r \trianglerighteq s, root(s) \in \mathcal{D}, l \not\triangleright v \sim_E s\}$ is the set of $A\vee C$-dependency pairs ($A\vee C$-DPs) of $\mathcal{R}$.*

Requiring $l \not\triangleright v \sim_E s$ for $\mathsf{DP}_{AC}(\mathcal{R})$ in Definition 4 follows Dershowitz's criteria [6] extended to $A\vee C$ rewrite theories . In general, the set of $A\vee C$-DPs which is obtained from Definition 4 is a subset of those which are obtained by particularizing Giesl and Kapur's definitions to the $A\vee C$ case [10].

Example 10 *Consider the AC-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 5. The set $DP_E(R)$ consists of the following pairs:*

$$
\begin{aligned}
F(x,x) &\to F(0, f(1,2)) && (3)\\
F(x,x) &\to F(1,2) && (4)\\
F(f(x,x),y) &\to F(f(0,f(1,2)),y) && (5)\\
F(f(x,x),y) &\to F(0,f(1,2)) && (6)\\
F(f(x,x),y) &\to F(1,2) && (7)
\end{aligned}
$$

6.1 Chains of $A\vee C$-DPs

An essential property of the dependency pair method is that it provides a *characterization* of termination of TRSs $\mathcal{R}$ as the absence of infinite (minimal) *chains of dependency pairs* [1, 13]. If we want to prove the same for $A\vee C$-rewrite theories, we have to introduce a suitable notion of chain which can be used with $A\vee C$-DPs. As in the DP-framework, where the origin of *pairs* does not matter, we should rather think of another rewrite theory $\mathcal{P} = (\Gamma, F, P)$ which is used together with $\mathcal{R}$ to build the chains. According to the usual terminology [13], we often call *pairs* to the rules $u \to v \in P$.

In the following definition, given sets of equations E and F, we let $\hat{=}_{F,E} = (\overset{\Lambda}{\vdash\!\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)^*$. Moreover, we define $\xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}}$ as the application of rules $l \to r \in \mathcal{S}$ such that $root(l) = f$.

Definition 5 (Chain of Pairs - Minimal Chain) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$- rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. An (F, P, E, R, S)-chain is a finite or infinite sequence of pairs $u_i \to v_i \in P$, together with substitutions σ and θ_i satisfying that, for all $i \geq 1$:*

1. *If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F$ or $v'_i = u'_i \in F$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then*

$$\sigma(v_i) \,\hat{=}_{F,E}\, \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \;\to^*_{\mathcal{E}xt_E(R),E} \circ \,\hat{=}_{F,E}\, \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \,\hat{=}_{F,E}\, \sigma(u_{i+1})$$

2. *and $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \,\hat{=}_{F,E}\, \sigma(u_{i+1})$, otherwise.*

An (F, P, E, R, S)-chain is called minimal *if for all $i \geq 1$, and $t'_i \,\hat{=}_{F,E}\, t_i$, t'_i is $(\mathcal{E}xt_E(R), E)$-terminating.*

As usual, in Definition 5 we assume that different occurrences of dependency pairs do not share any variable (renaming substitutions are used if necessary).

Note that the definition derives directly from Theorem 3: First we have to look for the minimal term of $\sigma(v_i)$, i.e. t_i, (see Theorem 2) which can be rewritten by using $\xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E}$ and again, since minimality can be lost we have to apply again Theorem 2 to connect with the next pair in the chain. This more abstract notion of chain can be particularized to be used with $A\vee C$-DPs, by just taking

1. $P = \mathsf{DP}_E(R)$,
2. $F = E^\sharp$, where $E^\sharp = \{s^\sharp = t^\sharp \mid s = t \in E\}$, and
3. $\mathcal{S} = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$.

We have the following:

Proposition 12 *Let Σ be a signature and E be a set of noncollapsing equations over Σ. Let $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, $s \sim_E t$ if and only if $s^\sharp \,\hat{=}_{E^\sharp,E}\, t^\sharp$.*

Proof. We have $s \sim_E t$ if and only if $s\ (\overset{\Lambda}{\vdash\!\!\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)^*\ t$. We proceed by induction on the length of the $(\overset{\Lambda}{\vdash\!\!\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)$-sequence from s to t. If $n = 0$, then $s = t$ and $s^\sharp = t^\sharp$. By reflexivity of $\hat{=}_{E^\sharp,E}$, we have $s^\sharp \,\hat{=}_{E^\sharp,E}\, t^\sharp$. If $n > 0$, then $s\ (\overset{\Lambda}{\vdash\!\!\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)\ s_0\ (\overset{\Lambda}{\vdash\!\!\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)^*\ t$ and, by the induction hypothesis, we know that $s_0^\sharp \,\hat{=}_{E^\sharp,E}\, t^\sharp$. Now, we consider two cases for the step $s\ (\overset{\Lambda}{\vdash\!\!\!\dashv}_E \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)\ s_0$:

1. If $s \overset{\Lambda}{\vdash\!\!\!\rightarrow}_E s_0$, then there is a substitution σ and an equation $u = v \in E$ such that $s = \sigma(u)$ and $s_0 = \sigma(v)$ or $s = \sigma(v)$ and $s_0 = \sigma(u)$. Therefore, since E is not collapsing, we have that $u, v \notin \mathcal{X}$. Then $s^\sharp = \sigma(u^\sharp)$ and $s_0^\sharp = \sigma(v^\sharp)$ (resp. $s^\sharp = \sigma(v^\sharp)$ and $s_0^\sharp = \sigma(u^\sharp)$). Therefore, since $u^\sharp = v^\sharp \in E^\sharp$, we have $s^\sharp \overset{\Lambda}{\vdash\!\!\!\rightarrow}_{E^\sharp} s_0^\sharp$. Hence, $s^\sharp \doteq_{E^\sharp, E} s_0$.

2. If $s \overset{>\Lambda}{\vdash\!\!\!\rightarrow}_E s_0$, then, since marking only affects the root symbol of s and s_0, we also have $s^\sharp \overset{>\Lambda}{\vdash\!\!\!\rightarrow}_E s_0^\sharp$. Hence, $s^\sharp \doteq_{E^\sharp, E} s_0$.

Thus, by transitivity of $\doteq_{E^\sharp, E}$, we conclude that $s^\sharp \doteq_{E^\sharp, E} t^\sharp$ as desired. We similarly prove that $s^\sharp \doteq_{E^\sharp, E} t^\sharp$ implies $s \sim_E t$. □

Proposition 13 *Let Σ be a signature $f \in \Sigma$ and $s, t \in \mathcal{T}(\Sigma, \mathcal{X})$. Then, $s \trianglerighteq_f t$ if and only if $s^\sharp \xrightarrow{\Lambda}{}^*_{\mathcal{S}_f} t^\sharp$.*

Theorem 4 (Soundness) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that*

$$S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

If there is no infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain, then $\mathcal{R}$ is $(\mathcal{E}xt_E(R), E)$-terminating

Proof. In the remainder of the proof, we let $F = E^\sharp$. We proceed by contradiction. If $\mathcal{R}$ is not $(\mathcal{E}xt_E(R), E)$-terminating, then, by Proposition 8, for each $(\mathcal{E}xt_E(R), E)$-nonterminating term there is an associated stably minimal term $s \in \mathcal{M}_{\infty, R, E}$. Let $f = root(s)$. We consider two cases for s.

1. If (1) $A_f \notin E_f$ or (2) $s = f(s_1, s_2)$, $A_f \in E_f$, and $root(s_1)$, $root(s_2) \in \mathcal{C}$, and for all $l \to r \in R_f$ and all subterms v of r ($r \trianglerighteq v$) such that $v = g(v_1, v_2)$ for some associative symbol g, we have that $root(v_1), root(v_2) \notin \mathcal{X} \cup \{g\}$. Then by Proposition 11, there exist $l \to r \in R$, a substitution σ and terms $t \in \mathcal{T}(\Sigma, \mathcal{X})$ and $u \in \mathcal{M}_{\infty, R, E}$ such that

$$s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R), E} t \sim_E \sigma(l) \xrightarrow{\Lambda}_R \sigma(r) \trianglerighteq u$$

and there is a nonvariable subterm v of r, $r \trianglerighteq v$, such that $u = \sigma(v)$. Hence, $u^\sharp = \sigma(v)^\sharp = \sigma(v^\sharp)$. By using Proposition 12 and, since $l^\sharp \to v^\sharp \in \mathsf{DP}(R)$ and $\mathsf{DP}(R) \subseteq \mathsf{DP}_E(R)$, we have

$$s^\sharp \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R), E} t^\sharp \doteq_{F, E} \sigma(l)^\sharp = \sigma(l^\sharp) \to_{\mathsf{DP}_E(R)} \sigma(v^\sharp) = u^\sharp$$

2. Otherwise, by Theorem 3, there is a rule $l \to r \in \mathcal{E}xt_E(R)$, a matching substitution σ and terms t'' and $u \in \mathcal{M}_{\infty, R, E}$ such that:

$$s \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R), E} t \sim_E t' \trianglerighteq_f t'' \sim_E \sigma(l) \xrightarrow{\Lambda}_{\mathcal{E}xt_E(R)} \sigma(r)$$

Furthermore, by Theorem 3, there is a nonvariable subterm v of r for which we have two possibilities:

(a) We have $v = g(v_1, v_2)$ for some associative symbol g, where $root(v_1) \in \mathcal{X} \cup \{g\}$ or $root(v_2) \in \mathcal{X} \cup \{g\}$, $root(\sigma(v_1)) = g$ or $root(\sigma(v_2)) = g$, $\sigma(v) \in \mathcal{T}_{\infty,R,E}$ and there is a term $w \sim_E \sigma(v)$ containing a strict g-subterm $u = g(u_1, u_2)$ $(w \rhd_g u)$ such that $u \in \mathcal{M}_{\infty,R,E}$. If we assume that there is an $v' \sim_E v$ which is a replacing subterm of l, i.e., $l \rhd v' \sim_E v$, then $\sigma(l) \rhd \sigma(v') \sim_E \sigma(v)$. Since $\sigma(v) \sim_E w \rhd_g u$ such that $u \in \mathcal{M}_{\infty,R,E}$, this contradicts that $\sigma(l) \in \mathcal{M}_{\infty,R,E}$. Thus, $l \not\rhd v' \sim_E v$. Since $l^\sharp \to v^\sharp \in \mathsf{DP}(\mathcal{E}xt_E(R))$ we have $\mathsf{DP}(\mathcal{E}xt_E(R)) \subseteq \mathsf{DP}_E(R)$. By using Propositions 12 and 13, we can write:

$$s^\sharp \xrightarrow{>\Lambda *}_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \circ \xrightarrow{\Lambda *}_{\mathcal{S}_f} t''^\sharp \doteq_{F,E} \sigma(l)^\sharp = \sigma(l^\sharp) \xrightarrow{\Lambda}_{\mathsf{DP}_E(R)} \sigma(v^\sharp) = \sigma(v)^\sharp \doteq_{F,E} w^\sharp \xrightarrow{\Lambda +}_{\mathcal{S}_f} u^\sharp$$

(b) Otherwise, $\sigma(v) \in \mathcal{M}_{\infty,R,E}$. If we assume that there is an $v' \sim_E v$ which is a replacing subterm of l, i.e., $l \rhd v' \sim_E v$, then $\sigma(l) \rhd \sigma(v') \sim_E \sigma(v)$. Since $\sigma(v) = u$ such that $u \in \mathcal{M}_{\infty,R,E}$, this contradicts that $\sigma(l) \in \mathcal{M}_{\infty,R,E}$. Thus, $l \not\rhd v' \sim_E v$. Hence, $l^\sharp \to v^\sharp \in \mathsf{DP}(\mathcal{E}xt_E(R)) \subseteq \mathsf{DP}_E(R)$, and, by using Propositions 12 and 13, we can write:

$$s^\sharp \xrightarrow{>\Lambda *}_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \circ \xrightarrow{\Lambda *}_{\mathcal{S}_f} t''^\sharp \doteq_{F,E} \sigma(l)^\sharp = \sigma(l^\sharp) \xrightarrow{\Lambda}_{\mathsf{DP}_E(R)} \sigma(v^\sharp) = u^\sharp$$

Note that, since $u \in \mathcal{M}_{\infty,R,E}$, we have that $u^\sharp$ is $(\mathcal{E}xt_E(R), E)$-terminating. Thus, $s^\sharp$ starts a minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain which could be infinitely extended from $u^\sharp$ in a similar way (as usual, in order to fit the requirement of variable-disjointness among two arbitrary pairs in a chain of pairs, we assume that appropriately renamed $A \vee C$-DPs are available when necessary). This contradicts our initial assumption. □

Now we prove that the previous $A \vee C$-dependency pairs approach is not only correct but also complete for proving $A \vee C$-termination.

Theorem 5 (Completeness) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that*

$$S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

If $\mathcal{R}$ is $(\mathcal{E}xt_E(R), E)$-terminating, then there is no infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain.

Proof. By contradiction. If there is an infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain, then there are substitutions σ_i and $A \vee C$-dependency pairs $u_i \to v_i \in \mathsf{DP}_E(R)$ such that :

1. If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F$ or $v'_i = u'_i \in F$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then

$$\sigma(v_i) \doteq_{F,E} \circ \xrightarrow{\Lambda}_{\mathcal{S}_{f_i}}^* t_i \to_{\mathcal{E}xt_E(R),E}^* \circ \doteq_{F,E} \circ \xrightarrow{\Lambda}_{\mathcal{S}_{f_i}}^* \circ \doteq_{F,E} \sigma(u_{i+1})$$

2. and $\sigma(v_i) = t_i \to_{\mathcal{E}xt_E(R),E}^* \circ \doteq_{F,E} \sigma(u_{i+1})$, otherwise.

Now, consider the first dependency pair $u_1 \to v_1$ in the sequence:

- If (1) holds, $u_1^\natural$ is the left-hand side of a rule $l_1 \to r_1 \in \mathcal{E}xt_E(R)$ and $v_1^\natural$ is a subterm of r_1. Therefore, $r_1 = C_1[v_1^\natural]_{p_1}$ for some $p_1 \in \mathcal{P}os_\Sigma(r_1)$ and we can perform the $A\vee C$-rewriting step $s_1 = \sigma_1(u_1^\natural) = \sigma_1(l_1) \to_{\mathcal{E}xt_E(R)} \sigma_1(r_1) = (C_1)[\sigma_1(v_1^\natural)]_{p_1}$, where, $\sigma_1(v_1^\natural)^\sharp = \sigma_1(v_1) \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_1}} t_1 \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \doteq_{F,E} \sigma_2(u_2)$ and $\sigma_2(u_2)$ initiates an infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain.

  By Theorem 3 we have that $t_1 \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} \circ \sim_E \circ \trianglerighteq_f \circ \sim_E s_2[\sigma_2(u_2^\natural)]_{p_1} = s_2$. Therefore, we can build in that way an infinite $A\vee C$-rewrite sequence

  $$s_1 \to_{\mathcal{E}xt_E(R)} \circ \sim_E \circ \trianglerighteq_f t_1 \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} \circ \sim_E \circ \trianglerighteq_f \circ \sim_E s_2 \to_{\mathcal{E}xt_E(R)} \cdots$$

  which contradicts the $(\mathcal{E}xt_E(R), E)$-termination of $\mathcal{R}$.

- If (2) holds, $u_1^\natural$ is the left-hand side of a rule $l_1 \to r_1 \in R$ and $v_1^\natural$ is a subterm of r_1. Therefore, $r_1 = C_1[v_1^\natural]_{p_1}$ for some $p_1 \in \mathcal{P}os_\Sigma(r_1)$ and we can perform the $A\vee C$-rewriting step $s_1 = \sigma_1(u_1^\natural) = \sigma_1(l_1) \to_R \sigma_1(r_1) = (C_1)[\sigma_1(v_1^\natural)]_{p_1}$, where, $t_1^\sharp = \sigma_1(v_1^\natural)^\sharp = \sigma_1(v_1) \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \sigma_2(u_2)$ and $\sigma_2(u_2)$ initiates an infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain.

  By Proposition 11 we have that $t_1 \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} \circ \sim_E s_2[\sigma_2(u_2^\natural)]_{p_1} = s_2$. Therefore, we can build in that way an infinite $A\vee C$-rewrite sequence

  $$s_1 \to_R t_1 \xrightarrow{>\Lambda}{}^*_{\mathcal{E}xt_E(R),E} \circ \sim_E s_2 \to_{\mathcal{E}xt_E(R)} \cdots$$

  which contradicts the $(\mathcal{E}xt_E(R), E)$-termination of $\mathcal{R}$.

□

As a corollary of Theorems 4 and 5, we have:

Corollary 5 (Characterization of $A\vee C$-Termination) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that $S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x, f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. Then, $\mathcal{R}$ is $(\mathcal{E}xt_E(R), E)$-terminating if and only if there is no infinite minimal $(E^\sharp, \mathsf{DP}_E(R), E, R, \mathcal{S})$-chain.*

7 An $A\vee C$-Dependency Pair Framework

In the following, we extend Giesl et al.'s DP-framework to provide a suitable framework for mechanizing proofs of $A\vee C$-termination using $A\vee C$-DPs.

Definition 6 ($A\vee C$ Problem) *An $A\vee C$ problem τ is a tuple $\tau = (F, P, E, R, S)$, where $\mathcal{R} = (\Sigma, E, R)$ is an $A\vee C$-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. An $A\vee C$ problem is finite if there is no infinite minimal (F, P, E, R, S)-chain. An $A\vee C$ problem τ is infinite if $\mathcal{R}$ is E-nonterminating or there is an infinite minimal (F, P, E, R, S)-chain.*

The following definition extends the notion of *DP-processor* [13] to prove termination of $A\vee C$-rewrite theories.

Definition 7 ($A\vee C$ Processor) *An $A\vee C$ processor* Proc *is a mapping from $A\vee C$ problems into sets of $A\vee C$ problems. Alternatively, it can also return* "no"*. An $A\vee C$ processor* Proc *is*

- sound *if for all $A\vee C$ problems τ, τ is finite whenever* $\mathsf{Proc}(\tau) \neq \mathsf{no}$ *and* $\forall \tau' \in \mathsf{Proc}(\tau)$*, τ' is finite.*
- complete *if for all $A\vee C$ problems τ, τ is infinite whenever* $\mathsf{Proc}(\tau) = \mathsf{no}$ *or* $\exists \tau' \in \mathsf{Proc}(\tau)$ *such that τ' is infinite.*

Similar to [13] for the DP-framework, we construct a tree whose nodes are labeled with $A\vee C$ problems or *"*yes*"* or *"*no*"*, and whose root is labeled with $(E^\sharp, \mathsf{DP}_E(R), E, R, S)$. Now we have the following result which extends [13, Corollary 5] to $A\vee C$-rewrite theories.

Theorem 6 ($A\vee C$-DP Framework) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-theory. We construct a tree whose nodes are labeled with $A\vee C$ problems or* "yes" *or* "no"*, and whose root is labeled with $(E^\sharp, \mathsf{DP}_E(R), E, R, S)$, where*

$$S = \{f^\sharp(f(x,y),z) \to f^\sharp(x,y), f^\sharp(x,f(y,z)) \to f^\sharp(y,z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

For every inner node labeled with τ, there is a sound processor Proc *satisfying one of the following conditions:*

1. $\mathsf{Proc}(\tau) = \mathsf{no}$ *and the node has just one child, labeled with* "no"*.*
2. $\mathsf{Proc}(\tau) = \varnothing$ *and the node has just one child, labeled with* "yes"*.*
3. $\mathsf{Proc}(\tau) \neq \mathsf{no}$*,* $\mathsf{Proc}(\tau) \neq \varnothing$*, and the children of the node are labeled with the $A\vee C$ problems in* $\mathsf{Proc}(\tau)$*.*

If all leaves of the tree are labeled with "yes"*, then $\mathcal{R}$ is E-terminating. Otherwise, if there is a leaf labeled with* "no" *and if all processors used on the path from the root to this leaf are complete, then $\mathcal{R}$ is not E-terminating.*

7.1 Preprocessing

A simply technique that can be useful when dealing with proofs of termination in the DP-framework is to try to remove rules from the original system before building the DP problem. In this way, we will start the proof with less rules and therefore less pairs, which can simplify the proof of termination. We extend here its use for proving E-termination. A reduction pair $(\gtrsim, \sqsupset)$ consists of a stable and monotonic quasi-ordering $\gtrsim$, and a stable and well-founded ordering $\sqsupset$ satisfying either $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$. We say that $(\gtrsim, \sqsupset)$ is monotonic if $\sqsupset$ is monotonic. $\sim$ is the stable, reflexive, transitive, and symmetric equivalence induced by $\gtrsim$, i.e., $\sim = \gtrsim \cap \lesssim$.

Proposition 14 (Removing strict rewrite rules) *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory. Let $(\gtrsim, \sqsupset)$ be a monotonic reduction pair such that $l\ (\gtrsim \cup \sqsupset)\ r$ for all $l \to r \in R$ and $u \sim v$ for all $u = v \in E$. Let $R_{\sqsupset} = \{l \to r \in R \mid l \sqsupset r\}$ and $R' = R - R_{\sqsupset}$. Then, $\mathcal{R}$ is E-terminating if and only if $\mathcal{R}' = (\Sigma, E, R')$ is E-terminating.*

Proof. Since $R' \subseteq R$, the *only if* part is obvious. For the *if* part, we proceed by contradiction. If $\mathcal{R}$ is not E-terminating, then there is an infinite E-rewrite sequence A:

$$t_1 \to_{R/E} t_2 \to_{R/E} \cdots t_n \to_{R/E} \cdots$$

that can be written in the following way:

$$t_1 \sim_E \circ \to_R \circ \sim_E t_2 \sim_E \circ \to_R \circ \sim_E \cdots t_n \sim_E \circ \to_R \circ \sim_E \cdots$$

where an infinite number of rules in $R_{\sqsupset}$ have been used; otherwise, there would be an infinite tail $t_m \to_{R'/E} t_{m+1} \to_{R'/E} \cdots$ for some $m \geq 1$ where only rules in R' are applied, contradicting the E-termination of $\mathcal{R}'$. Let $J = \{j_1, j_2, \ldots\}$ be the infinite set of indices indicating E-rewrite steps $t_j \to_{R/E} t_{j+1}$ in A, for all $j \in J$, where rules in $R_{\sqsupset}$ have been used to perform the E-rewriting step. Since $l \sqsupset r$ for all $l \to r \in R_{\sqsupset}$ and $u \sim v$ for all $u = v \in E$, by stability and monotonicity of $\sqsupset$ and $\sim$ (since $\sim = \gtrsim \cap \lesssim$), we have that $t_{j_i} \sqsupset t_{j_i+1}$. Since $l \gtrsim r$ for all $l \to r \in R'$, by stability and monotonicity of $\gtrsim$, we have that $t_{j_i+1} \gtrsim t_{j_{i+1}}$. By compatibility between $\gtrsim$ and $\sqsupset$ (and since $\sim = \gtrsim \cap \lesssim$), we have $t_{j_i} \sqsupset t_{j_{i+1}}$ for all $i \geq 1$. We obtain an infinite sequence $t_{j_1} \sqsupset t_{j_2} \sqsupset \cdots$ which contradicts well-foundedness of $\sqsupset$. □

7.2 $A \vee C$-Dependency Graph

$A \vee C$ problems focus our attention on the analysis of *infinite minimal chains*. Our aim here is obtaining a notion of graph which is able to represent all infinite *minimal* chains of pairs as given in Definition 5.

Definition 8 ($A \vee C$-Graph of Pairs) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$- rewrite theory and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. The $A \vee C$-*graph *associated to them (denoted $\mathsf{G}(F, P, E, R, S)$) has P as the set of nodes. There is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $u \to v, u' \to v'$ is an (F, P, E, R, S)-chain.*

In termination proofs, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [15]. A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle which is not contained in any other cycle. In the following result, given two sets of rules S and Q, we let S_Q be the least subset of S satisfying that whenever there is a rule $u \to v \in Q$, such that v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then $S_f \subseteq S_Q$.

Theorem 7 (SCC Processor) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$- rewrite theory and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. Then, the processor* Proc_{SCC} *given by*

$$\mathsf{Proc}_{SCC}(F, P, E, R, S) = \{(F, Q, E, R, S_Q) \mid Q \text{ are the pairs of an SCC in } \mathsf{G}(F, P, E, R, S)\}$$

is sound and complete.

As a consequence, we can *separately* work with the SCCs of $\mathsf{G}(F, P, E, R, S)$, disregarding other parts of the graph. Now we can use these notions to introduce the $A\vee C$-dependency graph, i.e., the $A\vee C$-graph whose nodes are the $A\vee C$-DPs instead of an arbitrary set of pairs.

Definition 9 ($A\vee C$-Dependency Graph) *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $\mathcal{S} = (\Sigma \cup \mathcal{D}^\sharp, S)$ be a TRS such that $S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}$. The* $A\vee C$-Dependency Graph *associated to $\mathcal{R}$ is:*

$$\mathsf{DG}(\mathcal{R}) = \mathsf{G}(E^\sharp, \mathsf{DP}_E(R), E, R, S)$$

7.3 Estimating the $A\vee C$-Dependency Graph

As in standard rewriting, the $A\vee C$-dependency graph of an $A\vee C$-rewrite theory is in general *not* computable. So, we need to use some approximation of it. For any term $t \in \mathcal{T}(\Sigma, \mathcal{X})$ let $\textsc{Cap}(t)$ result from replacing all proper subterms rooted by a defined symbol by fresh variables and let $\textsc{Ren}(t)$ which *independently* renames all *occurrences* of variables in t by using new fresh variables [1].

As usual, we should not talk about a *mgu* when dealing with rewriting modulo equations. Instead, the appropriate notion is that of complete set of E-unifiers. However, although in theory, all these E-unifiers have to be considered, for our results of reachability it is enough to check the existence of one.

Proposition 15 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A\vee C$-rewrite theory with $\Sigma = \mathcal{C} \uplus \mathcal{D}$. Let $u, t \in \mathcal{T}(\Sigma, \mathcal{X})$ be such that $\mathcal{V}ar(u) \cap \mathcal{V}ar(t) = \varnothing$ and θ, θ' be substitutions. If $\theta(t) \to^*_{\mathcal{E}xt_E(R),E} \circ \sim_E \theta'(u)$, then $\textsc{Ren}(\textsc{Cap}(t))$ and u E-unify.*

Proof. In the following, we let $s = \textsc{Ren}(\textsc{Cap}(t))$. Clearly, $t = \sigma(s)$ for some substitution σ. We proceed by induction on the length n of the sequence from $\theta(t) \to^*_{\mathcal{E}xt_E(R),E} t'$ in $\theta(t) \to^*_{\mathcal{E}xt_E(R),E} t' \sim_E \theta'(u)$.

1. If $n = 0$, then $\theta(t) = t' \sim_E \theta'(u)$. Since $t = \sigma(s)$, we have $\theta(\sigma(s)) \sim_E \theta'(u)$. Since $\mathcal{V}ar(s) \cap \mathcal{V}ar(u) = \varnothing$, we conclude that s and u E-unify.

2. If $n > 0$, then we have $t \to_{\mathcal{E}xt_E(R),E} t'' \to^*_{\mathcal{E}xt_E(R),E} t' \sim_E \theta'(u)$.

   Let $p \in \mathcal{P}os(t)$ be the position where the E-rewrite step $t \to_{\mathcal{E}xt_E(R),E} t''$ is performed. By definition of $\textsc{Cap}$ and $\textsc{Ren}$ we have that $s = s[z]_q$ for some (fresh) variable z and position q such that $q \leq p$. Let $s' = \textsc{Ren}(\textsc{Cap}(t''))$. Since $t'' = \sigma'(s')$ for some substitution σ', by the induction hypothesis,

$s'' = \text{REN}(\text{CAP}(s'))$ (which is just a renaming of the fresh variables in s', i.e., $s'' = \rho(s')$ for some renaming substitution ρ for such fresh variables) and u E-unify, i.e., there is a substitution ν such that $\nu(s'') \sim_E \nu(u)$. Note that we can write $s' = \tau(s)$ for some substitution τ such that $\tau(x) = x$ for all $x \neq z$ and $\tau(z) = s'|_q$. Therefore, $\nu(\rho(\tau(s))) \sim_E \nu(u)$, i.e., s and u E-unify.

□

Now, we are ready to provide a correct estimation of our graph of pairs. Correctness of our definition relies on Proposition 15.

Definition 10 (Estimated $A \vee C$-Graph of Pairs) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$- rewrite theory and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. The* estimated $A \vee C$-graph *associated to them (denoted $\mathsf{EG}(F, P, E, R, S)$) has P as the set of nodes and arcs which connect them as follows:*

1. *If v unifies with s for some $s = t \in F$ or $t = s \in F$ such that $s = f(s_1, s_2)$ and $s_1 \notin \mathcal{X}$ or $s_2 \notin \mathcal{X}$, then, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $root(u') = f$.*
2. *Otherwise, there is an arc from $u \to v \in P$ to $u' \to v' \in P$ if $\text{REN}(\text{CAP}(v))$ and u' $(F \cup E)$-unify (where equations in F can only be applied at root position).*

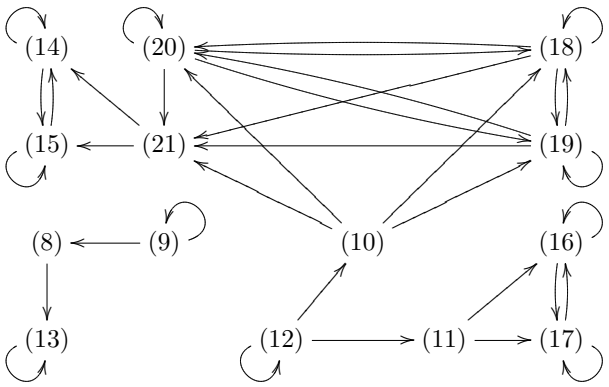
According to Definition 8, we would have the corresponding one for the *estimated* $A \vee C$-DG: $\mathsf{EDG}(\mathcal{R}) = \mathsf{EG}(E^\sharp, \mathsf{DP}_E(R), E, R, S)$, where

$$S = \{f^\sharp(f(x, y), z) \to f^\sharp(x, y), f^\sharp(x, f(y, z)) \to f^\sharp(y, z) \mid f \in \Sigma_A \cup \Sigma_{AC}\}.$$

Example 11 *For the A∨C-rewrite theory in Figure 1, the set* $\mathsf{DP}_E(R)$ *is*[1]*:*

$$\begin{aligned}
\texttt{LIST2SET}(\texttt{cons}(N,L)) &\to \texttt{UNION}(N,\texttt{list2set}(L)) && (8)\\
\texttt{LIST2SET}(\texttt{cons}(N,L)) &\to \texttt{LIST2SET}(L) && (9)\\
\texttt{IN}(N,\texttt{union}(M,S)) &\to \texttt{EQ}(N,M) && (10)\\
\texttt{IN}(N,\texttt{union}(M,S)) &\to \texttt{OR}(\texttt{eq}(N,M),\texttt{in}(N,S)) && (11)\\
\texttt{IN}(N,\texttt{union}(M,S)) &\to \texttt{IN}(N,S) && (12)\\
\texttt{UNION}(\texttt{union}(N,N),Z) &\to \texttt{UNION}(N,Z) && (13)\\
\texttt{AND}(\texttt{and}(\texttt{true},B),Z) &\to \texttt{AND}(B,Z) && (14)\\
\texttt{AND}(\texttt{and}(\texttt{false},B),Z) &\to \texttt{AND}(\texttt{false},Z) && (15)\\
\texttt{OR}(\texttt{or}(\texttt{true},B),Z) &\to \texttt{OR}(\texttt{true},Z) && (16)\\
\texttt{OR}(\texttt{or}(\texttt{false},B),Z) &\to \texttt{OR}(B,Z) && (17)\\
\texttt{EQ}(\texttt{s}(N),\texttt{s}(M)) &\to \texttt{EQ}(N,M) && (18)\\
\texttt{EQ}(\texttt{cons}(N,L),\texttt{cons}(M,L')) &\to \texttt{EQ}(N,M) && (19)\\
\texttt{EQ}(\texttt{cons}(N,L),\texttt{cons}(M,L')) &\to \texttt{EQ}(L,L') && (20)\\
\texttt{EQ}(\texttt{cons}(N,L),\texttt{cons}(M,L')) &\to \texttt{AND}(\texttt{eq}(N,M),\texttt{eq}(L,L')) && (21)
\end{aligned}$$

The (estimated) A∨C-DG is:



By using Theorem 7 we transform the A∨C problem $(E^\sharp, \mathsf{DP}(R), E, R, S)$ *into a set of A∨C problems* $\mathsf{Proc}_{SCC}(E^\sharp, \mathsf{DP}(R), E, R, S)$ *given by*

$$\{(E^\sharp,\{(9)\},E,R,\varnothing),(E^\sharp,\{(12)\},E,R,\varnothing),(E^\sharp,\{(13)\},E,R,S_{union}),$$
$$(E^\sharp,\{(14),(15)\},E,R,S_{and}),(E^\sharp,\{(16),(17)\},E,R,S_{or}),(E^\sharp,\{(18),(19),(20)\},E,R,\varnothing)\}$$

which contains six new (but simpler) A∨C problems.

[1] We have introduced new 'prefix' symbols *eq*, *cons* and *union* instead of the original 'infix' ones `_==_`, `_;_`, `__`.

7.4 Use of Reduction Pairs

In the dependency pair framework reduction pairs are used to obtain *smaller* sets of pairs $\mathcal{P}' \subseteq \mathcal{P}$ by removing the *strict* pairs, i.e., those pairs $u \to v \in \mathcal{P}$ such that $u \sqsupset v$. Stability is required both for $\gtrsim$ and $\sqsupset$ because, although we only check the left- and right-hand sides of the rewrite rules $l \to r$ (with $\gtrsim$) and pairs $u \to v$ (with $\gtrsim$ or $\sqsupset$), the chains of pairs involve *instances* $\sigma(l)$, $\sigma(r)$, $\sigma(u)$, and $\sigma(v)$ of rules and pairs and we aim at concluding $\sigma(l) \gtrsim \sigma(r)$, and $\sigma(u) \gtrsim \sigma(v)$ or $\sigma(u) \sqsupset \sigma(v)$, respectively. Monotonicity is required for $\gtrsim$ to deal with the application of rules $l \to r$ to an arbitrary depth in terms. Since the pairs are 'applied' only at the root level, no monotonicity is required for $\sqsupset$ (but, for this reason, we cannot compare the rules in $\mathcal{R}$ using $\sqsupset$). Dealing with associative and/or commutative axioms, we will compare them with the equivalence relation defined by the stable, reflexive, transitive, and symmetric equivalence $\sim$ induced by $\gtrsim$, i.e., $\sim = \gtrsim \cap \lesssim$, since we need to impose compatibility with the equational theories E and F. The following theorem formalizes a generic processor to remove pairs from $\mathcal{P}$ by using reduction pairs.

Theorem 8 (Reduction Pair Processor) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS. Let $(\gtrsim, \sqsupset)$ be a reduction pair such that*

1. *$R \subseteq \gtrsim$,*
2. *$P \cup S \subseteq \gtrsim \cup \sqsupset$, and*
3. *$E \cup F \subseteq \sim$.*

Let $P_\sqsupset = \{u \to v \in P \mid u \sqsupset v\}$ and $S_\sqsupset = \{s \to t \in S \mid s \sqsupset t\}$. Then, the processor Proc_{RP} given by

$$\mathsf{Proc}_{RP}(F, P, E, R, S) = \begin{cases} \{(F, P - P_\sqsupset, E, R, S - S_\sqsupset)\} & \text{if (1), (2), and (3) hold} \\ \{(F, P, E, R, S)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. Since $P - P_\sqsupset \subseteq P$ and $S - S_\sqsupset \subseteq S$, completeness is assured. Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal (F, P, E, R, S)-chain A, but that there is no infinite minimal $(F, P - P_\sqsupset, E, R, S - S_\sqsupset)$-chain. Due to the finiteness of P and S, we can assume that there is $Q \subseteq P$ and $T \subseteq S$ such that A has a tail B where all pairs in Q and rules in T are infinitely often used. We distinguish two kinds of elementary steps in B, according to Definition 5.

1. If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F$ or $v'_i = u'_i \in F$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then

$$\sigma(v_i) \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \doteq_{F,E} \sigma(u_{i+1})$$

Note that, due to the requirements imposed for the rules in R and S and equations in E and F, and by stability and transitivity of $\gtrsim$ (hence of $\sim$), monotonicity and transitivity of $\gtrsim$, we have

$$\sigma(v_i) \ \sim \circ \, (\gtrsim \cup \sqsupseteq) \, t_i \ \gtrsim \circ \sim \circ \, (\gtrsim \cup \sqsupseteq) \circ \sim \sigma(u_{i+1})$$

Here, it is important to specifically consider the case when the rules $l \to r$ involved in $\to^*_{\mathcal{E}xt_E(R),E}$-steps are taken from $\mathcal{E}xt_E(R) - R$, i.e, $l \to r \notin R$. In this case, we do not have an explicit compatibility requirement of $l \to r$ with $\gtrsim$, i.e., $l \gtrsim r$ is not explicitly required. However, since $\mathcal{R}$ is an $A \vee C$ theory, such rules are connected with rules rule $l' \to r' \in R$ in a simple way. For instance if $l = f(l', w) \to f(r', w) = r$ for some $l' \to r' \in R$ such that $root(l') = f$, then, since $l' \gtrsim r'$ holds, by monotonicity of $\gtrsim$, we also have $l = f(l', w) \gtrsim f(r', w) = r$. With other rules included in $\mathcal{E}xt_E(R) - R$ (see Section 3.1) we would proceed in a similar way. Now, taking into account that $\sim \circ \, (\gtrsim \cup \sqsupseteq) = \gtrsim \cup \sqsupseteq$ and $\sim \circ \gtrsim = \gtrsim$, we have

$$\sigma(v_i) \, (\gtrsim \cup \sqsupseteq) \, t_i \, (\gtrsim \cup \sqsupseteq) \sigma(u_{i+1})$$

Note that, by the compatibility condition required for $\gtrsim$ and $\sqsupseteq$, this means that $\sigma(v_i) \gtrsim \sigma(u_{i+1})$ or $\sigma(v_i) \sqsupseteq \sigma(u_{i+1})$.

2. If $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \sigma(u_{i+1})$, then we analogously have $\sigma(v_i) \gtrsim \sigma(u_{i+1})$.

Since $u_i \, (\gtrsim \cup \sqsupseteq) \, v_i$ for all $u_i \to v_i \in Q \subseteq P$, by stability of $\gtrsim$ and $\sqsupseteq$, we have $\sigma(u_i) \, (\gtrsim \cup \sqsupseteq) \, \sigma(v_i)$ for all $i \geq 1$. No pair $u \to v \in Q$ satisfies that $u \sqsupseteq v$ and no rule $s \to t \in T$ satisfies $s \sqsupseteq t$. Since $u \to v$ and $s \to t$ occurs infinitely often in B, and taking into account that $\sigma(v_i) \gtrsim \sigma(u_{i+1})$ or $\sigma(v_i) \sqsupseteq \sigma(u_{i+1})$ for all $i \geq 1$, there would be an infinite set $\mathcal{I} \subseteq \mathbb{N}$ such that $\sigma(u_i) \ \sqsupseteq \ \sigma(u_{i+1})$ for all $i \in \mathcal{I}$ or there would be an infinite set $\mathcal{J} \subseteq \mathbb{N}$ such that $\sigma(s_j) \ \sqsupseteq \ \sigma(t_{j+1})$ for all $j \in \mathcal{J}$. And we have $\sigma(u_i) \, (\gtrsim \cup \sqsupseteq) \, \sigma(u_{i+1})$ for all other $u_i \to v_i \in Q$ or $\sigma(s_j) (\gtrsim \cup \sqsupseteq) \sigma(t_{j+1})$ for all other $s_j \to t_j \in T$. Thus, by using the compatibility conditions of the reduction pair, we obtain an infinite decreasing $\sqsupseteq$-sequence which contradicts well-foundedness of $\sqsupseteq$.

Therefore, $Q \subseteq (P - P_{\sqsupseteq})$ and $T \subseteq (S - S_{\sqsupseteq})$, which means that B is an infinite minimal $(F, P - P_{\sqsupseteq}, E, R, S - S_{\sqsupseteq})$-chain, thus leading to a contradiction. □

7.5 Other Processors

Many times, the set of F axioms can be reduced to those equations that are really involved in minimal $A \vee C$-chains. The following processor shows a trivial method to eliminate them.

Theorem 9 (F Usable Equations Processor) *Let $\mathcal{P} = (\Gamma, F, P)$ be a rewrite theory, $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ be a TRS such that*

1. $root(u), root(v) \in \Gamma - \Sigma$ *for all* $u \to v \in P$,
2. $root(s) = root(t) \in \Gamma - \Sigma$ *for all* $s = t \in F$, *and*
3. $root(l) = root(r) \in \Gamma - \Sigma$ *for all* $l \to r \in S$, *and*

Let $\hat{F} = \{s = t \in F \mid root(s) = root(u)$ *or* $root(s) = root(v)$ *for some* $u \to v \in P\}$

Then, the processor Proc_{FUEq} *given by*

$$\mathsf{Proc}_{FUEq}(F, P, E, R, S) = \{(\hat{F}, P, E, R, S)\}$$

is sound and complete.

Proof. Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal (F, P, E, R, S)-chain A, but that there is no infinite minimal $(\hat{F}, P, E, R, S)$-chain. Due to the finiteness of P, we can assume that there is $Q \subseteq P$ and $F' \subseteq F$ such that A has a tail B where all pairs in Q and equations in F' are infinitely often used. We distinguish two kinds of elementary steps in B, according to Definition 5.

1. If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F'$ or $v'_i = u'_i \in F'$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then

   $$\sigma(v_i) \doteqdot_{F',E} t_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t'_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteqdot_{F',E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \doteqdot_{F',E} \sigma(u_{i+1})$$

   For this sequence we have:

   - $root(v_i) = f_i \in \Gamma - \Sigma$ (by (1)),
   - that means that in the step $\sigma(v_i) \doteqdot_{F',E} t'_i$ we can apply equations below the root by using E and if we apply an equation $s = t \in F'$, then $root(s) = root(t) = root(v_i) = f_i$ (by (2)) since we only use them at root position. Then, $s = t \in \hat{F}$.
   - In the step $t_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t'_i$, again we proceed in a similar way. Since for all $l \to r \in S$ we know that $root(l) = root(r)$ (by (3)), then we have that $root(t_i) = root(t'_i) = root(v_i)$.
   - The application of $\to^*_{\mathcal{E}xt_E(R),E}$-steps are below the root (since $root(t'_i) \in \Gamma - \Sigma$) and therefore the root symbol remains untouched.
   - In the next steps, since the root symbol remains unchanged proceeding like in previous steps, again, if a equation $s = t \in F'$ is applied at the root position has to be such that $root(s) = root(t) = root(v_i) = root(t_i) = root(t'_i) = \ldots = root(u_{i+1})$, therefore $s = t \in \hat{F}$.

2. If $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteqdot_{F',E} \sigma(u_{i+1})$, then we analogously have that the equations that can be used to connect with the next pair u_{i+1} are the equations in E and those from F' such that $root(s) = root(t) = root(v_i) = root(u_{i+1})$. Then, $s = t \in \hat{F}$.

Therefore, $root(v_i) = root(t_i) = root(t'_i) = \ldots = root(u_{i+1}) = f_i$ and $F' \subseteq \hat{F}$, which means that B is an infinite $(\hat{F}, P, E, R, S)$-chain. Since $\{s_i \mid s_i \trianglelefteq_{\hat{F},E} t_i\} \subseteq \{s_i \mid s_i \trianglelefteq_{F,E} t_i\}$ and by minimality, for all $w \trianglelefteq_{F,E} t_i$, w is $(\mathcal{E}xt_E(R), E)$-terminating therefore for all $w \trianglelefteq_{\hat{F},E} t_i$, w is $(\mathcal{E}xt_E(R), E)$-terminating. Therefore B is an infinite minimal $(\hat{F}, P, E, R, S)$-chain thus leading to a contradiction.

Regarding completeness, we proceed by contradiction. Assume that there is an infinite minimal $(\hat{F}, P, E, R, S)$-chain A, but that there is no infinite minimal (F, P, E, R, S)-chain. Due to the finiteness of P, we can assume that there is $Q \subseteq P$ and $F' \subseteq \hat{F}$ such that A has a tail B where all pairs in Q and equations in F' are infinitely often used. Since $\hat{F} \subseteq F$, every infinite $(\hat{F}, P, E, R, S)$-chain is also an infinite (F, P, E, R, S)-chain. Reasoning as in the correctness part over the infinite sequence, we know that $root(v_i) = root(t_i) = root(t'_i) = \ldots = root(u_{i+1}) = f_i$ and therefore, we conclude that the only equations from F that can be used in the infinite (F, P, E, R, S)-chain belong to $\hat{F}$. By minimality, for all $w(\overset{\Lambda}{\vdash\!\!\!\dashv}_{\hat{F}} \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)^* t_i$, w is $(\mathcal{E}xt_E(R), E)$-terminating, since the only equations that can be applied to t_i are those $\{s = t \in F \mid root(s) = root(t_i)$ *or* $root(t) = root(t_i)\}$ which correspond with $\hat{F}$, we can conclude that for all $w(\overset{\Lambda}{\vdash\!\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\!\dashv}_E)^* t_i$, w is $(\mathcal{E}xt_E(R), E)$-terminating. Therefore B is an infinite minimal (F, P, E, R, S)-chain thus leading to a contradiction.

□

Example 12 *By Example 11 we have* $\tau_0 = (E^\sharp, \mathsf{DP}(R), E, R, S)$ *by applying the SCC processor into* $\mathsf{Proc}_{SCC}(\tau_0) = \{\tau_1, \tau_2, \tau_3, \tau_4, \tau_5, \tau_6\}$ *where*

- $\tau_1 = (E^\sharp, \{(9)\}, E, R, \varnothing)$,
- $\tau_2 = (E^\sharp, \{(12)\}, E, R, \varnothing)$,
- $\tau_3 = (E^\sharp, \{(13)\}, E, R, S_{union})$,
- $\tau_4 = (E^\sharp, \{(14), (15)\}, E, R, S_{and})$,
- $\tau_5 = (E^\sharp, \{(16), (17)\}, E, R, S_{or})$ *and*
- $\tau_6 = (E^\sharp, \{(18), (19), (20)\}, E, R, \varnothing)$.

For the each of these A∨C problem,we can apply Proc_{FUEq}.

- *For* τ_1, *we have* $\mathsf{Proc}_{FUEq}(\tau_1) = (\varnothing, \{(9)\}, E, R, \varnothing)$,
- *For* τ_2, *we have* $\mathsf{Proc}_{FUEq}(\tau_2) = (\varnothing, \{(12)\}, E, R, \varnothing)$,
- *For* τ_3, *we have* $\mathsf{Proc}_{FUEq}(\tau_3) = (E^\sharp_{union}, \{(13)\}, E, R, S_{union})$,
- *For* τ_4, *we have* $\mathsf{Proc}_{FUEq}(\tau_4) = (E^\sharp_{and}, \{(14), (15)\}, E, R, S_{and})$,
- *For* τ_5, *we have* $\mathsf{Proc}_{FUEq}(\tau_5) = (E^\sharp_{or}, \{(16), (17)\}, E, R, S_{or})$ *and*
- *For* τ_6, *we have* $\mathsf{Proc}_{FUEq}(\tau_6) = (E^\sharp_{eq}, \{(18), (19), (20)\}, E, R, \varnothing)$.

8 Usable Rules and Equations for $A \vee C$ Problems

Usable rules are widely used in the DP framework to improve the power of DP processors. In this section we show how to obtain the set of usable rules and usable equations for a given $A \vee C$ problem and how to use them to define a new reduction pair processor. We follow the approach and techniques developed in [14, 26]. We assume that all all our rewrite theories are finite (they have no infinite rules or equations). Our first intuition was to define a proper notion of $A \vee C$-dependency that not only take into account the symbols occurring in the rules, but also the symbols occurring in the equations. But, since $A \vee C$ equations do not introduce new symbols in their left- and right-hand sides, we can use the standard notion of dependency that only considers symbols occurring in the rules. We use some auxiliar definitions. Let $Rls_R(f) = \{l \to r \in R \mid root(l) = f\}$, $Eqs_E(f) = \{u = v \in E \mid root(u) = f \vee root(v) = f\}$. Let $Fun(t) = \{f \mid \exists p \in \mathcal{P}os_{\mathcal{F}}(t), f = root(t|_p)\}$.

Definition 11 (Dependency [28]) *Let $\mathcal{R} = (\Sigma, R)$ be a TRS. We say that $f \in \Sigma$ has a* dependency *on $h \in \Sigma$ (written $f \rhd_R h$) if $f = h$ or there is a function symbol g with $g \rhd_R h$ and a rule $l \to r \in Rls_R(f)$ with $g \in Fun(r)$.*

To obtain the correct notions of usable rule and equation, we have to look at the structure of the chains. We have two possible ways to proceed in an (F, P, E, R, S)-chain. Given $u_i \to v_i \in P$ either

$$\sigma(v_i) \backsimeq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \;\to^*_{\mathcal{E}xt_E(R),E} \circ \backsimeq_{F,E} \circ \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} \circ \backsimeq_{F,E} \sigma(u_{i+1})$$

or

$$\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \backsimeq_{F,E} \sigma(u_{i+1})$$

Then, to obtain the set of usable rules and also usable equations we have to look for usable symbols not only in P, but also in F and S.

Definition 12 ($A \vee C$-Usable Rules and Equations) *Let τ be an $A \vee C$ problem such that $\tau = (F, P, E, R, S)$ where $\mathcal{R} = (\Sigma, E, R)$ is an $A \vee C$-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. The set $\mathcal{U}_R(\tau)$ of* $A \vee C$-usable rules *of τ is*

$$\begin{aligned} \mathcal{U}_R(\tau) \;=\; & \bigcup_{s \to t \,\in\, P, f \,\in\, Fun(t), f \rhd_R g} Rls_R(g) \;\cup \\ & \bigcup_{u = v \,\in\, F, f \,\in\, Fun(u) \,\cup\, Fun(v), f \rhd_R g} Rls_R(g) \;\cup \\ & \bigcup_{l \to r \,\in\, S, f \,\in\, Fun(r), f \rhd_R g} Rls_R(g) \end{aligned}$$

The set $\mathcal{U}_E(\tau)$ *of* $A\vee C$-usable equations *of* τ *is*

$$\begin{array}{rcll} \mathcal{U}_E(\tau) & = & \bigcup_{s\to t\,\in\,P, f\,\in\,Fun(t), f\triangleright_R g} Eqs_E(g) & \cup \\ & & \bigcup_{u=v\,\in\,F, f\,\in\,Fun(u)\,\cup\,Fun(v), f\triangleright_R g} Eqs_E(g) & \cup \\ & & \bigcup_{l\to r\,\in\,S, f\,\in\,\cup Fun(r), f\triangleright_R g} Eqs_E(g) & \end{array}$$

Note that, if the rules from S are of the form $f^\sharp(f(x,y),z) \to f^\sharp(x,y)$ or $f^\sharp(x,f(y,z)) \to f^\sharp(y,z)$ do not introduce new rules as usable.

Now, we define an interpretation that, given an $A\vee C$ problem $\tau = (F,P,E,R,S)$, allows us to transform any infinite minimal (F,P,E,R,S)-chain into an infinite sequence of pairs from P where, for all $i \geq 1$,

$$\sigma'(v_i) \,\hat{=}_{F,E'} \circ \to^*_{\mathcal{S}_{f_i}\cup\mathcal{C}_\varepsilon} t_i \;\to^*_{\mathcal{E}xt_{E'}(R'),E'} \circ \,\hat{=}_{F,E'} \circ \to^*_{\mathcal{S}_{f_i}\cup\mathcal{C}_\varepsilon} \circ \,\hat{=}_{F,E'} \circ \to^*_{\mathcal{C}_\varepsilon} \sigma'(u_{i+1})$$

or

$$\sigma'(v_i) = t_i \to^*_{\mathcal{E}xt_{E'}(R'),E'} \circ \,\hat{=}_{F,E'} \circ \to^*_{\mathcal{C}_\varepsilon} \sigma'(u_{i+1})$$

with $\mathcal{C}_\varepsilon = \{\mathtt{c}(x,y) \to x, \mathtt{c}(x,y) \to y\}$ being $\mathtt{c}$ a new fresh binary symbol, $E' = \mathcal{U}_E(\tau)$ and $R' = \mathcal{U}_R(\tau) \cup \mathcal{C}_\varepsilon$. We modify the original interpretation used in [14, 26] in such a way that, if a term is rooted by a non-usable $A\vee C$ symbol, then all its equivalent terms have exactly the same interpretation.

Definition 13 ($A\vee C$-Interpretation) *Let* $\mathcal{R} = (\Sigma, E, R)$ *be an* $A\vee C$*-rewrite theory and* $\Delta \subseteq \Sigma$*. Let* $>$ *be an arbitrary total ordering over* $\mathcal{T}(\Sigma \cup \{\bot, \mathtt{c}\}, \mathcal{X})$ *where* $\bot$ *is a fresh constant symbol and* $\mathtt{c}$ *is a fresh binary symbol. The* $A\vee C$*-Interpretation* $\mathcal{I}_{\Delta,E}$ *is a mapping from* E*-terminating terms in* $\mathcal{T}(\Sigma,\mathcal{X})$ *to terms in* $\mathcal{T}(\Sigma \cup \{\bot, \mathtt{c}\}, \mathcal{X})$ *defined as follows:*

$$\mathcal{I}_{\Delta,E}(t) = \begin{cases} t & \text{if} \quad t \in \mathcal{X} \\ \mathtt{f}(\mathcal{I}_{\Delta,E}(t_1), \ldots, \mathcal{I}_{\Delta,E}(t_k)) & \text{if} \quad t = \mathtt{f}(t_1,\ldots,t_k) \text{ and } \mathtt{f} \in \Delta \\ order(\{\mathtt{c}(\mathtt{f}(\mathcal{I}_{\Delta,E}(s_1), \ldots, \mathcal{I}_{\Delta,E}(s_k)), s') \mid s = \mathtt{f}(s_1,\ldots,s_n) \in [t]_E\}) & \text{if} \quad t = \mathtt{f}(t_1,\ldots,t_k) \text{ and } \mathtt{f} \notin \Delta \end{cases}$$

where $s' = order\left(\{\mathcal{I}_{\Delta,E}(u) \mid s \to_{\mathcal{E}xt_E(R),E} u\}\right)$

$$order(T) = \begin{cases} \bot, & \text{if } T = \varnothing \\ \mathtt{c}(t, order(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

We have to ensure now that the adapted interpretation does not generate infinite terms. This is achieved thanks to the fact that for any $A\vee C$-equational theory E, E-equivalence classes are always finite, and reductions with $\to_{\mathcal{E}xt_E(R),E}$ are finitely branching due to finiteness of R (assumed in Section 3).

Lemma 1 *Let* $\mathcal{R} = (\Sigma, E, R)$ *be an* $A\vee C$*-rewrite theory and* $\Delta \subseteq \Sigma$ *and* $t \in \mathcal{T}(\Sigma,\mathcal{X})$*. If* t *is* E*-terminating then* $\mathcal{I}_{\Delta,E}$ *is well-defined.*

PROOF. According to Definition 13, to obtain an infinite term as result of $\mathcal{I}_{\Delta,E}(t)$, either: (1) we get a term t' such that $[t']_E$ is infinite, or (2) we would have to perform an infinite number of applications of the function *order* $(\{\mathcal{I}_{\Delta,E}(u) \mid s \to_{\mathcal{R}} u\})$:

(1) We know that $\mathcal{R}$ is an $A \vee C$-rewrite theory. Therefore all the equations are of the form $f(f(x,y),z) = f(x,f(y,z))$ or $f(x,y) = f(y,x)$. Since equations are linear and no new symbols are added, $[t']_E$ is finite.

(2) We have an infinite sequence of the following way:

$$t = t_0 \sim_E t_1 \to_{\mathcal{E}xt_E(R),E} t_2 \sim_E t_3 \to_{\mathcal{E}xt_E(R),E} \cdots$$

that contradicts the E-termination of t.

□

Now, to prove the main theorem, we need some auxiliar results that allow us to construct the new infinite sequence. The idea is that, for each relation in the chain R, if $s \; R \; t$ then $\mathcal{I}_{\Delta,E}(s) \; R \; \mathcal{I}_{\Delta,E}(t)$.

Definition 14 *Let $\mathcal{R} = (\Sigma, E, R)$ be a rewrite theory, $\Delta \subseteq \Sigma$ and σ be a substitution. We define $\sigma_{\mathcal{I}_{\Delta,E}}$ as $\sigma_{\mathcal{I}_{\Delta,E}}(x) = \mathcal{I}_{\Delta,E}(\sigma(x))$.*

Lemma 2 *Let $\mathcal{R} = (\Sigma, E, R)$ be an $A \vee C$-rewrite theory and $\Delta \subseteq \Sigma$. Let t be a term and σ be a substitution. If $\sigma(t)$ is E-terminating, then $\mathcal{I}_{\Delta,E}(\sigma(t)) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(t)$. If t only contains Δ symbols, then $\mathcal{I}_{\Delta,E}(\sigma(t)) = \sigma_{\mathcal{I}_{\Delta,E}}(t)$.*

PROOF. By structural induction on t:

- If $t = x$ is a variable then $\mathcal{I}_{\Delta,E}(\sigma(x)) = \sigma_{\mathcal{I}_{\Delta,E}}(x)$.
- If $t = f(t_1, \ldots, t_k)$ then
  - If $f \in \Delta$ then $\mathcal{I}_{\Delta,E}(\sigma(t)) = f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k)))$. By hypothesis, terms $\sigma(t_i)$ are E-terminating for $1 \leq i \leq ar(f)$. By induction hypothesis, for all terms t_i we have $\mathcal{I}_{\Delta,E}(\sigma(t_i)) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(t_i)$. This implies $f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k))) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(t)$.
  - If $f \notin \Delta$, we have that for all $s = f(s_1, \ldots, s_n) \in [t]_E$ we obtain $\mathcal{I}_{\Delta,E}(\sigma(t)) \to^+_{\mathcal{C}_\varepsilon} c(f(\mathcal{I}_{\Delta,E}(\sigma(s_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(s_k))), s'))$ using proper $\mathcal{C}_\varepsilon$-steps. Since $t \in [t]_E$, we can obtain $\mathcal{I}_{\Delta,E}(\sigma(t)) \to^+_{\mathcal{C}_\varepsilon} c(f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k))), t'))$. Therefore, we get $f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k)))$ applying again a $\mathcal{C}_\varepsilon$ rule. Then, we conclude that $f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k))) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(t)$ reasoning as in the previous item.

Therefore, we have that $\mathcal{I}_{\Delta,E}(\sigma(t)) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(t)$.

The second part of the lemma is proved similarly. By structural induction on t:

- If $t = x$ is a variable then $\mathcal{I}_{\Delta,E}(\sigma(x)) = \sigma_{\mathcal{I}_{\Delta,E}}(x)$.
- If $t = f(t_1, \ldots, t_k)$ and $f \in \Delta$ (because t only contains Δ symbols), then $\mathcal{I}_{\Delta,E}(\sigma(t)) = f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k)))$. By hypothesis, terms $\sigma(t_i)$ are E-terminating for $1 \leq i \leq ar(f)$ and terms t_i only contain Δ symbols. Therefore, by induction hypothesis, $\mathcal{I}_{\Delta,E}(\sigma(t_i)) = \sigma_{\mathcal{I}_{\Delta,E}}(t_i)$. This implies $f(\mathcal{I}_{\Delta,E}(\sigma(t_1)), \ldots, \mathcal{I}_{\Delta,E}(\sigma(t_k))) = \sigma_{\mathcal{I}_{\Delta,E}}(t)$.

□

Lemma 3 *Let $\tau = (F, P, E, R, S)$ be an A∨C problem where $\mathcal{R} = (\Sigma, E, R)$ is an A∨C-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau))$ or $v = u \in (E - \mathcal{U}_E(\tau))\})$. If s and t are E-terminating and $s \overset{\Lambda}{\vdash\!\!\dashv}_F t$ then $\mathcal{I}_{\Delta,E}(s) \overset{\Lambda}{\vdash\!\!\dashv}_F \mathcal{I}_{\Delta,E}(t)$.*

Proof. Let $s \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} t$ and $s = \sigma(u) \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} \sigma(v) = t$ or $s = \sigma(v) \overset{\Lambda}{\vdash\!\!\dashv}_{\{v=u\}} \sigma(u) = t$ for some substitution σ. Since $u, v \in \mathcal{T}(\Delta, \mathcal{X})$ by the construction of Δ, by Lemma 2 we get $\mathcal{I}_{\Delta,E}(\sigma(u)) = \sigma_{\mathcal{I}_{\Delta,E}}(u) \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} \sigma_{\mathcal{I}_{\Delta,E}}(v) = \mathcal{I}_{\Delta,E}(\sigma(v))$ or $\mathcal{I}_{\Delta,E}(\sigma(v)) = \sigma_{\mathcal{I}_{\Delta,E}}(v) \overset{\Lambda}{\vdash\!\!\dashv}_{\{v=u\}} \sigma_{\mathcal{I}_{\Delta,E}}(u) = \mathcal{I}_{\Delta,E}(\sigma(u))$. □

Lemma 4 *Let $\tau = (F, P, E, R, S)$ be an A∨C problem where $\mathcal{R} = (\Sigma, E, R)$ is an A∨C-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau))$ or $v = u \in (E - \mathcal{U}_E(\tau))\})$. If s and t are E-terminating and $s \overset{p}{\vdash\!\!\dashv}_E t$ then $\mathcal{I}_{\Delta,E}(s) \overset{p\ *}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.*

Proof. We proceed by induction on the position $p \in \mathcal{P}os(s)$ of the redex in the reduction $s \overset{p}{\vdash\!\!\dashv}_{\{u=v\}} t$.

- First, we consider that $root(s) \in \Delta$.
  - If $p = \Lambda$ (therefore $u = v \in \mathcal{U}_E(\tau)$). So we have $s = \sigma(u) \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} \sigma(v) = t$ or $s = \sigma(v) \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} \sigma(u) = t$ for some substitution σ. Moreover, $u, v \in \mathcal{T}(\Delta, \mathcal{X})$ by the construction of Δ. By Lemma 2, we get $\mathcal{I}_{\Delta,E}(\sigma(u)) = \sigma_{\mathcal{I}_{\Delta,E}}(u) \overset{\Lambda}{\vdash\!\!\dashv}_{\{u=v\}} \sigma_{\mathcal{I}_{\Delta,E}}(v) = \mathcal{I}_{\Delta,E}(\sigma(v))$ or $\mathcal{I}_{\Delta,E}(\sigma(v)) = \sigma_{\mathcal{I}_{\Delta,E}}(v) \overset{\Lambda}{\vdash\!\!\dashv}_{\{v=u\}} \sigma_{\mathcal{I}_{\Delta,E}}(u) = \mathcal{I}_{\Delta,E}(\sigma(u))$.
  - If $p \neq \Lambda$ then $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $t = f(s_1, \ldots, t_i, \ldots, s_n)$, $s_i \overset{q}{\vdash\!\!\dashv}_{\{u=v\}} t_i$ and $p = i.q$. By the induction hypothesis, $\mathcal{I}_{\Delta,E}(s_i) \overset{q\ *}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t_i)$, $\mathcal{I}_{\Delta,E}(s) \overset{i.q\ *}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$ and, hence $\mathcal{I}_{\Delta,E}(s) \overset{p\ *}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.

- Finally, we consider the case $root(s) \notin \Delta$. Since we can infer that $t \in [s]_E$ and $[s]_E = [t]_E$ because $s \overset{p}{\vdash\!\!\dashv}_{\{u=v\}} t$ then, $\mathcal{I}_{\Delta,E}(s) = \mathcal{I}_{\Delta,E}(t)$ and, hence, $\mathcal{I}_{\Delta,E}(s) \overset{p}{\vdash\!\!\dashv}^*_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.

□

Lemma 5 *Let $\tau = (F, P, E, R, S)$ be an $A \vee C$ problem where $\mathcal{R} = (\Sigma, E, R)$ is an $A \vee C$-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau))$ or $v = u \in (E - \mathcal{U}_E(\tau))\})$. If s and t are E-terminating and $s \doteq_{F,E} t$ then $\mathcal{I}_{\Delta,E}(s) \doteq_{F,\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.*

Proof. We can write $s \doteq_{F,E} t$ as $s(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_E)^* t$ and we know:

1. If $s' = t'$ trivially $\mathcal{I}_{\Delta,E}(s') = \mathcal{I}_{\Delta,E}(t')$.
2. If $s' \overset{\Lambda}{\vdash\!\!\dashv}_F t'$ for any two terms s', t' then $\mathcal{I}_{\Delta,E}(s') \overset{\Lambda}{\vdash\!\!\dashv}_F \mathcal{I}_{\Delta,E}(t')$ by Lemma 3.
3. If $s' \overset{>\Lambda}{\vdash\!\!\dashv}_E t'$ for any two terms s', t' then $\mathcal{I}_{\Delta,E}(s') \overset{>\Lambda^*}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t')$ by Lemma 4.
4. If $s' \overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_E t'$ for any two terms s', t' then $\mathcal{I}_{\Delta,E}(s') \overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda^*}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)}$ $\mathcal{I}_{\Delta,E}(t')$ by (2) and (3), which is equivalent to $\mathcal{I}_{\Delta,E}(s')(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)})^* \mathcal{I}_{\Delta,E}(t')$.

Now, we proceed on the length n of the sequence $s(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_E)^* t$.

- If $n = 0$ then $s = t$ and $\mathcal{I}_{\Delta,E}(s) = \mathcal{I}_{\Delta,E}(t)$.
- If $n > 0$ then $s(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_E) u (\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_E)^* t$. By the induction hypothesis, $\mathcal{I}_{\Delta,E}(u)(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)})^* \mathcal{I}_{\Delta,E}(t)$, and by (4) we have $\mathcal{I}_{\Delta,E}(s)(\overset{\Lambda}{\vdash\!\!\dashv}_F \cup \overset{>\Lambda}{\vdash\!\!\dashv}_{\mathcal{U}_E(\tau)})^* \mathcal{I}_{\Delta,E}(u)$.

Hence, if $s \doteq_{F,E} t$ then $\mathcal{I}_{\Delta,E}(s) \doteq_{F,\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$. □

Lemma 6 *Let $\tau = (F, P, E, R, S)$ be an $A \vee C$ problem where $\mathcal{R} = (\Sigma, E, R)$ is an $A \vee C$-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau))$ or $v = u \in (E - \mathcal{U}_E(\tau))\})$. If s and t are E-terminating and $s \to_{\mathcal{E}xt_E(R),E} t$ then $\mathcal{I}_{\Delta,E}(s) \to^+_{\mathcal{E}xt_{\mathcal{U}_E(\tau)}(\mathcal{U}_R(\tau) \cup \mathcal{C}_\varepsilon),\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.*

Proof. We proceed by induction on the position $p \in \mathcal{P}os(s)$ of the redex in the reduction $s \overset{p}{\sim}_E s' \overset{p}{\longrightarrow}_{l \to r} t$ where $l \to r \in \mathcal{E}xt_E(R)$. By recursively applying Lemma 4 to $s = s_1 \vdash\!\!\dashv_E s_2 \vdash\!\!\dashv_E \cdots \vdash\!\!\dashv_E s_n = s'$, we have that $\mathcal{I}_{\Delta,E}(s) = \mathcal{I}_{\Delta,E}(s_1) \vdash\!\!\dashv_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(s_2) \vdash\!\!\dashv_{\mathcal{U}_E(\tau)} \cdots \vdash\!\!\dashv_{\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(s_n) = \mathcal{I}_{\Delta,E}(s')$.

- First, let $root(s) = root(s') \in \Delta$.
  - If $p = \Lambda$ $(l \to r \in \mathcal{E}xt_{\mathcal{U}_E(\tau)}(\mathcal{U}_R(\tau)))$, we have $s' = \sigma(l) \xrightarrow{\Lambda}_{\{l\to r\}} \sigma(r) = t$ for some substitution σ. Moreover, $r \in \mathcal{T}(\Delta, \mathcal{X})$ by the construction of Δ. By Lemma 2, we get
  $$\mathcal{I}_{\Delta,E}(\sigma(l)) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(l) \to_{\{l\to r\}} \sigma_{\mathcal{I}_{\Delta,E}}(r) = \mathcal{I}_{\Delta,E}(\sigma(r))$$
  - If $p \neq \Lambda$ then $s = f(s_1, \ldots, s_i, \ldots, s_n)$, $t = f(s_1, \ldots, t_i, \ldots, s_n)$ and $s_i \to_{\{l\to r\},E} t_i$. By the induction hypothesis,
  $$\mathcal{I}_{\Delta,E}(s_i) \to^+_{\mathcal{E}xt_{\mathcal{U}_E(\tau)}(\mathcal{U}_R(\tau)\cup\mathcal{C}_\varepsilon),\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t_i)$$
  and, hence also $\mathcal{I}_{\Delta,E}(s) \to^+_{\mathcal{E}xt_{\mathcal{U}_E(\tau)}(\mathcal{U}_R(\tau)\cup\mathcal{C}_\varepsilon),\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$.
- Finally, let $root(s) = root(s') \notin \Delta$. Then,
  $$\mathcal{I}_{\Delta,E}(t) \in order\left(\{\mathcal{I}_{\Delta,E}(u) \mid s \to_{\mathcal{E}xt_E(R),E} u\}\right)$$
  because $s \to_{\mathcal{E}xt_E(R),E} t$. By applying $\mathcal{C}_\varepsilon$ rules, we get $\mathcal{I}_{\Delta,E}(s) \to^+_{\mathcal{C}_\varepsilon} \mathcal{I}_{\Delta,E}(t)$.

Therefore, $\mathcal{I}_{\Delta,E}(s) \to^*_{\mathcal{E}xt_{\mathcal{U}_E(\tau)}(\mathcal{U}_R(\tau)\cup\mathcal{C}_\varepsilon),\mathcal{U}_E(\tau)} \mathcal{I}_{\Delta,E}(t)$. □

Lemma 7 *Let $\tau = (F, P, E, R, S)$ be an A∨C problem where $\mathcal{R} = (\Sigma, E, R)$ is an A∨C-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau))$ or $v = u \in (E - \mathcal{U}_E(\tau))\})$. If s and t are E-terminating and $s \xrightarrow{\Lambda}_{S_{f_i}} t$ then $\mathcal{I}_{\Delta,E}(s) \to^+_{S_{f_i}\cup\mathcal{C}_\varepsilon} \mathcal{I}_{\Delta,E}(t)$.*

Proof. We know that $p = \Lambda$ and $l \to r \in S_{f_i}$. So we have $s = \sigma(l) \xrightarrow{\Lambda}_{\{l\to r\}} \sigma(r) = t$ for some substitution σ. Moreover, $r \in \mathcal{T}(\Delta, \mathcal{X})$ by the construction of Δ. By Lemma 2 we get $\mathcal{I}_{\Delta,E}(\sigma(l)) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(l) \to_{\{l\to r\}} \sigma_{\mathcal{I}_{\Delta,E}}(r) = \mathcal{I}_{\Delta,E}(\sigma(r))$. Therefore, $\mathcal{I}_{\Delta,E}(s) \to^+_{S_{f_i}\cup\mathcal{C}_\varepsilon} \mathcal{I}_{\Delta,E}(t)$. □

A relation $\gtrsim$ is $\mathcal{C}_\varepsilon$-compatible iff $\mathtt{c}(x, y) \gtrsim x$ and $\mathtt{c}(x, y) \gtrsim y$ for a new binary fresh symbol $\mathtt{c}$.

Theorem 10 (RP Processor with A∨C-Usable Rules and Equations) *Let $\tau = (F, P, E, R, S)$ be an A∨C problem where $\mathcal{R} = (\Sigma, E, R)$ is an A∨C-rewrite theory, $\mathcal{P} = (\Gamma, F, P)$ is a rewrite theory, and $\mathcal{S} = (\mathcal{F}, S)$ is a TRS. Let $(\gtrsim, \sqsupset)$ be a reduction pair such that $\gtrsim$ is $\mathcal{C}_\varepsilon$-compatible and*

1. $\mathcal{U}_R(\tau) \subseteq \gtrsim$,
2. $(P \cup S) \subseteq \gtrsim \cup \sqsupset$, *and*
3. $F \cup \mathcal{U}_E(E) \subseteq \sim$.

Let $P_{\sqsupset} = \{u \to v \in P \mid u \sqsupset v\}$ *and* $S_{\sqsupset} = \{s \to t \in S \mid s \sqsupset t\}$. *Then, the processor* Proc_{RP} *given by*

$$\mathsf{Proc}_{RP}(F, P, E, R, S) = \begin{cases} \{(F, P - P_{\sqsupset}, E, R, S - S_{\sqsupset})\} & \text{if (1), (2), and (3) hold} \\ \{(F, P, E, R, S)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Proof. Since $P - P_{\sqsupset} \subseteq P$ and $S - S_{\sqsupset} \subseteq S$, completeness is assured. Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal (F, P, E, R, S)-chain A, but that there is no infinite minimal $(F, P - P_{\sqsupset}, E, R, S - S_{\sqsupset})$-chain. Due to the finiteness of P and S, we can assume that there is $Q \subseteq P$ and $T \subseteq S$ such that A has a tail B where all pairs in Q and rules in T are infinitely often used. We distinguish two kinds of elementary steps in B, according to Definition 5.

1. If $\sigma(v_i) = f_i(v_{i1}, v_{i2})$ satisfies $\sigma(v_i) = \theta_i(u'_i)$ for some $u'_i = v'_i \in F$ or $v'_i = u'_i \in F$ such that $u'_i = f_i(u'_{i1}, u'_{i2})$ satisfies $u'_{i1} \notin \mathcal{X}$ or $u'_{i2} \notin \mathcal{X}$, then

$$\sigma(v_i) \doteq_{F,E} s_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i \to^*_{\mathcal{E}xt_E(R),E} t'_i \doteq_{F,E} w_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} w'_i \doteq_{F,E} \sigma(u_{i+1})$$

We apply $\mathcal{I}_{\Delta,E}$ in Definition 13 to the initial term in the sequence. We let $\Delta = (\Gamma \cup \Sigma \cup \mathcal{F}) - (\{root(l) \mid l \to r \in (R - \mathcal{U}_R(\tau))\} \cup \{root(u) \mid u = v \in (E - \mathcal{U}_E(\tau)) \text{ or } v = u \in (E - \mathcal{U}_E(\tau))\})$, $E' = \mathcal{U}_E(\tau)$ and $R' = \mathcal{U}_R(\tau) \cup \mathcal{C}_\varepsilon$. Sequentially, we obtain the following results:

- Since v_i only contains Δ symbols, by Lemma 2 we have that $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) = \mathcal{I}_{\Delta,E}(\sigma(v_i))$
- By Lemma 5, $\mathcal{I}_{\Delta,E}(\sigma(v_i)) \doteq_{F,E'} \mathcal{I}_{\Delta,E}(s_i)$.
- By induction on the length of the sequence $s_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} t_i$ and using Lemma 7, $\mathcal{I}_{\Delta,E}(s_i) \to^*_{S_{f_i} \cup \mathcal{C}_\varepsilon} \mathcal{I}_{\Delta,E}(t_i)$.
- By induction on the length of the sequence $t_i \to^*_{\mathcal{E}xt_E(R),E} t'_i$ and using Lemma 6, $\mathcal{I}_{\Delta,E}(t_i) \to^*_{\mathcal{E}xt_{E'}(R'),E'} \mathcal{I}_{\Delta,E}(t'_i)$.
- By Lemma 5, $\mathcal{I}_{\Delta,E}(t'_i) \doteq_{F,E'} \mathcal{I}_{\Delta,E}(w_i)$.
- By induction on the length of the sequence $w_i \xrightarrow{\Lambda}{}^*_{\mathcal{S}_{f_i}} w'_i$ and using Lemma 7, $\mathcal{I}_{\Delta,E}(w_i) \to^*_{S_{f_i} \cup \mathcal{C}_\varepsilon} \mathcal{I}_{\Delta,E}(w'_i)$.
- By Lemma 5, $\mathcal{I}_{\Delta,E}(w'_i) \doteq_{F,E'} \mathcal{I}_{\Delta,E}(\sigma(u_{i+1}))$.
- By Lemma 2, $\mathcal{I}_{\Delta,E}(\sigma(u_{i+1})) \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$.

Therefore, we obtain the following chain:

$$\begin{array}{c} \sigma_{\mathcal{I}_{\Delta,E}}(v_i) \doteq_{F,E'} \circ \to^*_{\mathcal{S}_{f_i} \cup \mathcal{C}_\varepsilon} \circ \to^*_{\mathcal{E}xt_{E'}(R'),E'} t''_i \\ t''_i \doteq_{F,E'} \circ \to^*_{\mathcal{S}_{f_i} \cup \mathcal{C}_\varepsilon} \circ \doteq_{F,E'} \circ \to^*_{\mathcal{C}_\varepsilon} \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1}) \end{array}$$

Note that, due to the requirements imposed for the rules in $\mathcal{U}_R(\tau)$ and S and equations in $\mathcal{U}_E(\tau)$ and F, and by stability, transitivity and $\mathcal{C}_\varepsilon$-compatibility of $\gtrsim$ (hence of $\sim$), monotonicity and transitivity of $\gtrsim$, we have

$$\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \sim \circ(\gtrsim \cup \sqsupset)\circ \ \gtrsim \circ \sim \circ(\gtrsim \cup \sqsupset)\circ \sim \circ \gtrsim \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$$

Here, it is important to specifically consider the case when the rules $l \to r$ involved in $\to_{\mathcal{E}xt_{E'}(R'),E'}$-steps are taken from $\mathcal{E}xt_{E'}(R') - (R')$, i.e, $l \to r \notin R'$. In this case, we do not have an explicit compatibility requirement of $l \to r$ with $\gtrsim$, i.e., $l \gtrsim r$ is not explicitly required. However, since $\mathcal{R}' = (\Sigma, E', R')$ is an $A\vee C$ rewrite theory, such rules are connected with rules rule $l' \to r' \in R'$ in a simple way. For instance if $l = f(l', w) \to f(r', w) = r$ for some $l' \to r' \in R'$ such that $root(l') = f$, then, since $l' \gtrsim r'$ holds, by monotonicity of $\gtrsim$, we also have $l = f(l', w) \gtrsim f(r', w) = r$. With other rules included in $\mathcal{E}xt_{E'}(R') - (R')$ (see Section 3.1) we would proceed in a similar way. Now, taking into account that $\sim \circ (\gtrsim \cup \sqsupset) = \gtrsim \cup \sqsupset$ and $\sim \circ \gtrsim = \gtrsim$, we have

$$\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \ (\gtrsim \cup \sqsupset) \ \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$$

Note that, by the compatibility condition required for $\gtrsim$ and $\sqsupset$, this means that $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \gtrsim \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$ or $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \sqsupset \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$.

2. If $\sigma(v_i) = t_i \to^*_{\mathcal{E}xt_E(R),E} \circ \doteq_{F,E} \sigma(u_{i+1})$, then we analogously, applying Lemma 6 and Lemma 5, have

$$\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \to^*_{\mathcal{E}xt_{E'}(R'),E'} \circ \doteq_{F,E'} \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$$

and, hence, $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \gtrsim \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$.

Since $u_i \ (\gtrsim \cup \sqsupset) \ v_i$ for all $u_i \to v_i \in Q \subseteq P$, by stability of $\gtrsim$ and $\sqsupset$, we have $\sigma_{\mathcal{I}_{\Delta,E}}(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma_{\mathcal{I}_{\Delta,E}}(v_i)$ for all $i \geq 1$. No pair $u \to v \in Q$ satisfies that $u \sqsupset v$, and no rule $s \to t \in T$ satisfies $s \sqsupset t$. Since $u \to v$ and $s \to t$ occurs infinitely often in B, and taking into account that $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \gtrsim \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$ or $\sigma_{\mathcal{I}_{\Delta,E}}(v_i) \sqsupset \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$ for all $i \geq 1$, there would be an infinite set $\mathcal{J} \subseteq \mathbb{N}$ such that $\sigma_{\mathcal{I}_{\Delta,E}}(u_i) \sqsupset \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$ for all $i \in \mathcal{J}$ or there would be an infinite set $\mathcal{K} \subseteq \mathbb{N}$ such that $\sigma_{\mathcal{I}_{\Delta,E}}(s_j) \sqsupset \sigma_{\mathcal{I}_{\Delta,E}}(t_{j+1})$ for all $j \in \mathcal{K}$. And we have $\sigma_{\mathcal{I}_{\Delta,E}}(u_i) \ (\gtrsim \cup \sqsupset) \ \sigma_{\mathcal{I}_{\Delta,E}}(u_{i+1})$ for all other $u_i \to v_i \in Q$, and $\sigma_{\mathcal{I}_{\Delta,E}}(s_j) \ (\gtrsim \cup \sqsupset) \ \sigma_{\mathcal{I}_{\Delta,E}}(t_{j+1})$ for all other $u_j \to v_j \in T$. Thus, by using the compatibility conditions of the reduction pair, we obtain an infinite decreasing $\sqsupset$-sequence which contradicts well-foundedness of $\sqsupset$.

Therefore, $Q \subseteq (P - P_\sqsupset)$ and $T \subseteq (S - S_\sqsupset)$, which means that B is an infinite chain, thus leading to a contradiction. □

Example 13 *For the $A \vee C$-rewrite theory in Figure 1, we have the following rules in R (with prefix symbols again):*

$$\begin{aligned}
\texttt{list2set}(N) &\to N && (22)\\
\texttt{list2set}(\texttt{cons}(N, L)) &\to \texttt{union}(N, \texttt{list2set}(L)) && (23)\\
\texttt{in}(N, \texttt{null}) &\to \texttt{false} && (24)\\
\texttt{in}(N, \texttt{union}(M, S)) &\to \texttt{or}(\texttt{eq}(N, M), \texttt{in}(N, S)) && (25)\\
\texttt{union}(N, N) &\to N && (26)\\
\texttt{and}(\texttt{true}, B) &\to B && (27)\\
\texttt{and}(\texttt{false}, B) &\to \texttt{false} && (28)\\
\texttt{or}(\texttt{true}, B) &\to \texttt{true} && (29)\\
\texttt{or}(\texttt{false}, B) &\to B && (30)\\
\texttt{eq}(\texttt{0}, \texttt{s}(N)) &\to \texttt{false} && (31)\\
\texttt{eq}(\texttt{s}(N), \texttt{s}(M)) &\to \texttt{eq}(N, M) && (32)\\
\texttt{eq}(\texttt{cons}(N, L), M) &\to \texttt{false} && (33)\\
\texttt{eq}(\texttt{cons}(N, L), \texttt{cons}(M, L')) &\to \texttt{and}(\texttt{eq}(N, M), \texttt{eq}(L, L')) && (34)\\
\texttt{eq}(L, L) &\to \texttt{true} && (35)
\end{aligned}$$

By example 12, we have the following $A \vee C$ problems:

- $\tau_1' = (\varnothing, \{(9)\}, E, R, \varnothing)$,
- $\tau_2' = (\varnothing, \{(12)\}, E, R, \varnothing)$,
- $\tau_3' = (E^\sharp_{union}, \{(13)\}, E, R, S_{union})$,
- $\tau_4' = (E^\sharp_{and}, \{(14), (15)\}, E, R, S_{and})$,
- $\tau_5' = (E^\sharp_{or}, \{(16), (17)\}, E, R, S_{or})$ *and*
- $\tau_6' = (E^\sharp_{eq}, \{(18), (19), (20)\}, E, R, \varnothing)$.

For the each of these $A \vee C$ problem,we can apply Proc_{UR}.

- *In the case of τ_1' we have:*
  $\mathcal{U}_R(\tau_1') = \varnothing$, $\mathcal{U}_E(\tau_1') = \varnothing$ *and the following polynomial interpretation*[2]*:*

$$[\texttt{LIST2SET}](x) = x * x + x \qquad [\texttt{cons}](x, y) = y + 1$$

  to conclude finiteness of τ_1'.

[2]The quasi-orderings $\gtrsim$ induced by a polynomial interpretation can always be made compatible with the rules of the TRS $\mathcal{C}_\varepsilon$, i.e., $\mathcal{C}_\varepsilon \subseteq \gtrsim$.

- *For τ_2' we have:*
  $\mathcal{U}_R(\tau_2') = \varnothing$, $\mathcal{U}_E(\tau_2') = \varnothing$ *and the following polynomial interpretation:*

$$[\texttt{IN}](x,y) = x*y+y \qquad [\texttt{union}](x,y) = y+1$$

  to conclude finiteness of τ_2'.

- *For τ_3' we have:*
  $\mathcal{U}_R(\tau_3') = \{(26)\}$, $\mathcal{U}_E(\tau_3') = \{(E_{union})\}$ *and the following polynomial interpretation:*

$$[\texttt{UNION}](x,y) = x+y+1 \qquad [\texttt{union}](x,y) = x+y$$

  to conclude finiteness of τ_3'.

- *For τ_4' we have:*
  $\mathcal{U}_R(\tau_4') = \{(27),(28)\}$, $\mathcal{U}_E(\tau_4') = \{(E_{and})\}$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{AND}](x,y) & = & x+y & & [\texttt{and}](x,y) & = & x+y+1 \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

  This processor eliminate one strict pair and generate a new $A \vee C$ problem $\tau_{4.1} = (E_{and}^{\sharp}, \{(14)\}, E, R, S_{and})$ where again we have:
  $\mathcal{U}_R(\tau_{4.1}) = \{(27),(28)\}$, $\mathcal{U}_E(\tau_{4.1}) = \{(E_{and})\}$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{AND}](x,y) & = & x+y & & [\texttt{and}](x,y) & = & x+y \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

  to conclude finiteness of $\tau_{4.1}$ and therefore of τ_4'.

- *For τ_5' we have:*
  $\mathcal{U}_R(\tau_5') = \{(29),(30)\}$, $\mathcal{U}_E(\tau_5') = \{(E_{or})\}$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{OR}](x,y) & = & x*y+x+y & & [\texttt{or}](x,y) & = & x*y+x+y \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

  This processor eliminate one strict pair and generate a new $A \vee C$ problem $\tau_{5.1} = (E_{or}^{\sharp}, \{(16)\}, E, R, S_{or})$ where again we have:

$\mathcal{U}_R(\tau_{5.1}) = \{(29), (30)\}$, $\mathcal{U}_E(\tau_{5.1}) = \{(E_{and})\}$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{OR}](x,y) & = & x+y & & [\texttt{or}](x,y) & = & x+y+1 \\ [\texttt{false}] & = & 1 & & [\texttt{true}] & = & 1 \end{array}$$

to conclude finiteness of $\tau_{5.1}$ *and therefore of* τ_5'.

- *Finally, for* τ_6' *we have:*
  $\mathcal{U}_R(\tau_6') = \varnothing$, $\mathcal{U}_E(\tau_6') = \varnothing$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{EQ}](x,y) & = & x*y+x+y & & [\texttt{cons}](x,y) & = & x+y+1 \\ [\texttt{s}](x) & = & x+1 & & & & \end{array}$$

This processor eliminate one strict pair and generate a new $A \vee C$ *problem* $\tau_{6.1} = (E_{eq}^{\sharp}, \{(18), (19)\}, E, R, \varnothing)$ *where we have:*
$\mathcal{U}_R(\tau_{6.1}) = \varnothing$, $\mathcal{U}_E(\tau_{6.1}) = \varnothing$ *and the following polynomial interpretation:*

$$\begin{array}{rclcrcl} [\texttt{EQ}](x,y) & = & x*y+x+y & & [\texttt{cons}](x,y) & = & x+1 \\ [\texttt{s}](x) & = & x+1 & & & & \end{array}$$

This application eliminate another strict pair and generate a new $A \vee C$ *problem* $\tau_{6.2} = (E_{eq}^{\sharp}, \{(18)\}, E, R, \varnothing)$. *We have:*
$\mathcal{U}_R(\tau_{6.2}) = \varnothing$, $\mathcal{U}_E(\tau_{6.2}) = \varnothing$ *and the following polynomial interpretation:*

$$[\texttt{EQ}](x,y) = x*y+x+y \qquad [\texttt{s}](x) = x+1$$

to conclude finiteness of $\tau_{6.2}$ *and therefore of* τ_6'.

Therefore, after showing the finiteness of all the $A \vee C$ *problems generated from Example 1, we can conclude its E-termination.*

9 Benchmarks

We have implemented all techniques described in this paper in the termination tool MU-TERM. MU-TERM is a tool which can be used to verify a number of termination properties of (variants of) Term Rewriting Systems (TRSs): termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and, thanks to this new approach, termination of rewriting modulo specific axioms. With these new features implemented, MU-TERM has been able to participate in the International Competition

of Termination Tools[3] in the category of *TRS Equational*. This is not the first implementation for proving termination of rewriting modulo axioms: C*i*ME [5] is able to prove AC-termination of TRSs, and AProVE [11] is able to deal with termination of rewriting modulo equations satisfying some restrictions. However, in the last editions of the competition C*i*ME has not participated and AProVE is the only termination tool that participates in this category from its first edition in 2004. There exists a *Termination Problem Data Base*[4] (TPDB) which contains 71 examples in the equational category[5]. In the 2010 edition there were only two participants: AProVE and MU-TERM. The organization selected randomly a subset of 34 examples from the entire set. MU-TERM was able to solve 16 out of them whereas AProVE solved 24. We considered this result as a good one since only a few techniques had been implemented to deal with termination modulo axioms and AProVE implements specific techniques since 2004. These include an *AC*-recursive path order (RPO) with status (3 examples out of them are solved with it) and processors based on usable rules (the remaining 5 examples are solved using them). There is no formal publication of any of these techniques. In the case of the AC-RPO, we suppose that they implement the master thesis of Stephan Falke [9] although [24] was published before. Recently, we have found out that this work was adapted to the dependency pair framework in the master thesis of Christian Stein ([25], in german and not available publicly). However, both papers are based on the notion of minimality presented in [10] which we have shown that is not appropriate. In the case of processors for managing usable rules is essential to deal with a correct notion of minimality [14, 26].

Now, with only the techniques described in this paper, MU-TERM is able to solve 59 examples out of 71. Two examples more than AProVE[6]. For full details see:

`http://zenon.dsic.upv.es/muterm/benchmarks/benchmarks-avc/benchmarks.html`

In comparison with the implementation of the techniques developed in [3], where MU-TERM were able to solve 39 examples[7], now, thanks to the new techniques, MU-TERM has become a powerful and competitive tool for proving termination of $A \vee C$-rewrite theories. The practical results are summarized in Table 1.

10 Related Work and Conclusions

This paper is an extended and revised version of [3]. We provide complete proofs for all results, and also present more examples about the use of the theory. The main conceptual differences between [3] and this paper can be summarized as follows:

[3] See `http://www.lri.fr/~marche/termination-competition/`

[4] See `http://termination-portal.org/wiki/TPDB`

[5] We have used version 7.0.2 of the TPDB.

[6] See the 2008 edition of the termination competition where the entire set of examples from the category where considered.

[7] see `http://www.dsic.upv.es/~balarcon/WRLA10/benchmarks.html`

	MU-TERM $A \vee C$-DPs	APROVE	MU-TERM [3]
YES score	59	57	39
YES average time	6.83 sec.	5.12 sec.	40.13 sec.

Table 1: Comparative in proofs of termination of $A \vee C$-rewrite theories

- We have refined the notion of $A \vee C$- dependency pairs integrating an equational extension of Dershowitz's refinement of standard dependency pairs (see [6]).
- We have refined the notion of $A \vee C$-chain by allowing the application of F axioms only at the root position.
- We have developed a preprocessing technique which is often able to remove rules from the original system before starting the proof in the $A \vee C$ DP-framework, thus simplifying the whole proof of $A \vee C$-termination.
- We have refined the $A \vee C$ processor of reduction pairs which is now able to eliminate rules, not only from R, but also from the set S.
- We have developed a new $A \vee C$ processor that restricts the set of F axioms to those that are really used in the $A \vee C$ problem.
- We have extended the well-known technique of usable rules to $A \vee C$-termination and we have developed the corresponding $A \vee C$ processor to eliminate pairs and rules by means of reduction orders.
- We have implemented the techniques presented in [3] and the ones developed here. We have made some benchmarks showing the performance of them.

As remarked in the introduction, this is not the first work which tries to use dependency pairs for proving termination of rewriting modulo an equational theory, see [9, 10, 16, 17, 18, 20, 21, 25]. Our work, however, is, as far as we know, the first one which provides a satisfactory notion of minimal non-terminating term for an $A \vee C$-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ which can be used to provide a suitable definition of minimal chain of dependency pairs, which can in turn be used to characterize $A \vee C$-termination (Corollary 5). In order to substantiate this claim, consider the AC-rewrite theory $\mathcal{R} = (\Sigma, E, R)$ in Example 5 again. The $A \vee C$-DPs for $\mathcal{R}$ are enumerated in Example 10. Such dependency pairs coincide with the ones which would be computed by, e.g., [9, 10, 17, 18, 25]. Remember that t in Example 5 is *minimal* in Giesl and Kapur's sense (Definition 2); and also according to [9, 25] which inherit this notion. We should, then, be able to find an infinite *minimal* chain of DPs starting from $t^\sharp$. According to [9, 10, 17, 18, 25], 'minimal' means that $\sigma(v_i)$ is $(\mathcal{E}xt_E(R), E)$-terminating for all pairs $u_i \to v_i \in \mathsf{DP}_E(R)$ in the chain of dependency pairs induced by the substitution σ. However, this is *not* possible:

the marked version $t^\sharp$ of t is $F(f(0,1),f(0,f(1,2)))$, which is an $(\mathcal{E}xt_E(R),E)$-terminating term. After some $E^\sharp \cup E$-equivalence steps (where $E^\sharp$ is applied only at root position) we would be able to apply one of the rules in $\mathsf{DP}_E(R)$. Note, however, that *no rule* $u \to v \in \mathsf{DP}_E(R)$ except (5) has a right-hand side v which can be rewritten (after instantiation into $\sigma(v)$) into an instance $\sigma(u')$ of the left-hand side u' of any other pair in $\mathsf{DP}_E(R)$ by means of $(\mathcal{E}xt_E(R), E^\sharp \cup E)$-rewriting steps. This means that only the dependency pair (5) could be used in any infinite minimal chain of dependency pairs starting from $t^\sharp$. But such a chain would start as follows:

$$F(f(0,1),f(0,f(1,2))) \doteq_{E^\sharp,E} F(f(0,0),f(1,f(1,2))) \to_{(5)} F(f(0,f(1,2)),f(1,f(1,2)))$$

where $F(f(0,f(1,2)),f(1,f(1,2)))$ contains a subterm $f(1,f(1,2))$ which, as showed in Example 5, is $(\mathcal{E}xt_E(R),E)$-nonterminating. Therefore, this chain of dependency pairs is *not* minimal. We conclude that, according to the notion of minimal chain in the aforementioned papers, *there is no minimal chain of pairs starting from* $t^\sharp$. This means that no *sound* approach to proving AC-termination on the basis of such notion of minimal chain is possible. In this paper we have introduced the notion of *stably minimal term* (Definition 3) which overcomes these problems (Proposition 11 and Theorem 3) and leads to an appropriate characterization of $A\vee C$-termination as the absence of infinite minimal chains of $A\vee C$-DPs (Definitions 4 and 5, and Corollary 5).

Furthermore, we note that [17, 18] deal with *AC-rewrite theories* only, and that [10], which considers more general rewrite theories E including $A\vee C$-theories do not cover our work in a second respect: when purely associative theories are considered (i.e., rewrite theories $\mathcal{R} = (\Sigma, E, R)$ such that $E_f \subseteq \{A_f\}$ for all $f \in \Sigma$), then Giesl and Kapur's technique requires the computation of *instances* of the rules in $\mathcal{E}xt_E(R)$ for which the computation of *all* the E-unifiers $uni_E(v,l)$ of v and l for the rules $l \to r$ in $\mathcal{E}xt_E(R)$ and equations $u = v \in E$ or $v = u \in E$ is required. It is well-known, however, that the E-unification problem for associative theories E is *infinitary*, which means that $uni_E(v,l)$ is not guaranteed to be finite, in general. In sharp contrast, we do not have to do that for dealing with purely associative rewrite theories $\mathcal{R}$.

Our second main (and novel) contribution is the formalization of an $A\vee C$-dependency pair framework (Definitions 6 and 7) which, on the basis of the previously developed theory, can be used to develop automatic tools for proving termination of $A\vee C$-rewrite theories (Theorem 6). Several important processors have been developed as well: the SCC processor (Theorem 7), the reduction pair processor (Theorem 8), the processor that restricts the set of F axioms (Theorem 9), and the reduction pair processor with usable rules and equations (Theorem 10). We have implemented the techniques described in this paper in the termination tool MU-TERM and we have developed some benchmarks, showing that our $A\vee C$-DP Framework is currently the most powerful approach for proving termination of $A\vee C$-rewrite theories. As we have commented, the implementation of the techniques in [3] allowed us to participate in the termination competition in the equational category in the TPDB and therefore

providing MU-TERM the ability of proving termination modulo axioms. Thanks to these new improvements, MU-TERM is a powerful tool for proving termination of $A \vee C$- rewrite theories and as far as we know, no tool is able to solve more examples from the equational category. Much work remains ahead in terms of further developing the new $A \vee C$-dependency pair framework. Appropriate reduction orderings which are well-suited for being used in the reduction pair processor should be investigated. It would also be very useful to explore how the requirements on E can be relaxed to handle even more general sets of axioms. Regarding tool support for the method we have presented, we have integrated it within the tool MU-TERM [2]. In this way, our termination technique modulo arbitrary combinations of associative and/or commutative axioms is applicable to an even wider range of rewrite theories, which can be transformed into $A \vee C$-theories by non-termination-preserving transformations [7, 8, 19].

References

[1] T. Arts, J. Giesl, Termination of Term Rewriting Using Dependency Pairs, Theoretical Computer Science 236(1–2) (2000) 133–178.

[2] B. Alarcón, R. Gutiérrez, S. Lucas and R. Navarro-Marset, Proving Termination Properties with MU-TERM, in: M. Johnson, D. Pavlovic (Eds.), Proc. of Thirteenth International Conference on Algebraic Methodology And Software Technology, AMAST'10, LNCS, vol. 6486, Springer, 2010, pp. 201–208.

[3] B. Alarcón, S. Lucas, J. Meseguer, A Dependency Pair Framework for $A \vee C$-Termination, in: P. Ölveczky (Ed.), Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10, LNCS, vol. 6381, Springer, 2010, pp. 35–51.

[4] F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, 1998.

[5] E. Contejean, C. Marché, B. Monate, X. Urbain, Proving Termination of Rewriting with C*i*ME, in: A. Rubio (Ed.), Proc. of the 6th International Workshop on Termination, WST'03, 2003, pp. 71–73.

[6] N. Dershowitz, Termination by Abstraction, in: B. Demoen and V. Lifschitz (Eds.), Proc. of 20th International Conference on Logic Programming, ICLP'04, LNCS, vol. 3132, Springer, 2004, pp. 1–18.

[7] F. Durán, S. Lucas, J. Meseguer, Termination Modulo Combinations of Equational Theories, in: S. Ghilardi, R. Sebastiani (Eds.), Proc of 7th International Symposium on Frontiers of Combining Systems, FroCoS'09, LNAI, vol. 5749, Springer, 2009, pp. 246–262.

[8] F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving Operational Termination of Membership Equational Programs, Higher-Order and Symbolic Computation 21(1-2) (2008) 59–88.

[9] S.P. Falke, Automated Termination Analysis for Equational Rewriting, Diplomarbeit, Fakultät für Informatik, Rheinisch-Westfälische Technische Hochschule Aachen (2004).

[10] J. Giesl, D. Kapur, Dependency Pairs for Equational Rewriting, in: A. Middeldorp (Ed.), 12th International Conference of Rewriting Techniques and Applications, RTA'01, LNCS, vol. 2051, Springer, 2001, pp. 93–108.

[11] J. Giesl, P. Schneider-Kamp, R. Thiemann, AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework, in U. Furbach, N. Shankar (Eds.), Proc. of Third International Joint Conference on Automated Reasoning, LNAI, vol. 4130, Springer, 2006, pp. 281–286.

[12] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. in: F. Baader, A. Voronkov (Eds.), Proc. of XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04, LNAI, vol. 3452, Springer, 2004, pp. 301–331.

[13] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Mechanizing and Improving Dependency Pairs, Journal of Automatic Reasoning, 37(3) (2006) 155–203.

[14] N. Hirokawa, A. Middeldorp, Dependency Pairs Revisited, in: V. van Oostrom (Ed.), Proc of 15th International Conference on Rewriting Techniques and Applications, RTA'04, LNCS, vol. 3091, Springer, 2004, pp. 249–268.

[15] N. Hirokawa, A. Middeldorp, Automating the Dependency Pair Method, Information and Computation 199 (2005) 172–199.

[16] K. Kusakari, Termination, AC-Termination and Dependency Pairs of Term Rewriting Systems, PhD. Thesis, School of Information Science, JAIST (2000).

[17] K. Kusakari, M. Nakamura, Y. Toyama, Elimination Transformations for Associative-Commutative Rewriting Systems, Journal of Automated Reasoning 37 (2006) 205–229.

[18] K. Kusakari, Y. Toyama, On Proving AC-Termination by AC-Dependency Pairs, IEICE Transactions on Information and Systems, E84-D, 604–612 (2001).

[19] S. Lucas, J. Meseguer, Operational Termination of Membership Equational Programs: the Order-Sorted Way, in: G. Rosu (Ed.), Proc. of the 7th International Workshop on Rewriting Logic and its Applications, WRLA'08, Electronic Notes in Theoretical Computer Science. 238(3), (2009), 207–225.

[20] C. Marché, X. Urbain, Termination of Associative-Commutative Rewriting by Dependency Pairs, in: T. Nipkow (Ed.), Proc of 9th International Conference on Rewriting Techniques and Applications, RTA'98, LNCS, vol. 1379, Springer, 1998, pp. 241–255.

[21] C. Marché, X. Urbain, Modular and Incremental Proofs of AC-Termination, Journal of Symbolic Computation 38 (2004) 873–897.

[22] E. Ohlebusch, Advanced Topics in Term Rewriting, Springer, 2002.

[23] G.E. Peterson, M.E. Stickel, Complete Sets of Reductions for Some Equational Theories, Journal of the ACM. 28(2) (1981) 233–264.

[24] A. Rubio. A Fully Syntactic AC-RPO. *Information and Computation*, 178(2): 515–533, 2002.

[25] C. Stein, Das Dependency Pair Framework zur automatischen Terminierungsanalyse von Termersetzung modulo Gleichungen, Diplomarbeit, Fakultät für Informatik, Rheinisch-Westfälische Technische Hochschule Aachen (2006).

[26] R. Thiemann, J. Giesl, P. Schneider-Kamp, Improved Modular Termination Proofs Using Dependency Pairs, in: D. Basin, M. Rusinowitch (Eds), Proc of 2nd International Joint Conference on Automated Reasoning, IJCAR'04, LNAI, vol. 3097, Springer, 2004, pp. 75–90.

[27] TeReSe, Term Rewriting Systems, Cambridge University Press, 2003.

[28] X. Urbain, Modular & Incremental Automated Termination Proofs, Journal of Automated Reasoning 32 (2004) 315–355.

19

Appendix A: Detailed Benchmarks on ICSR

Problem	MuTerm-ICSDPs	AProVe-Transf	AProVe-Transf	Problem
TRS/CSR/Ex1_2_AEL03	0,042	timeout	timeout	TRS/CSR/Ex1_2_AEL03_C
			timeout	TRS/CSR/Ex1_2_AEL03_GM
			timeout	TRS/CSR/Ex1_2_AEL03_iGM
TRS/CSR/Ex1_2_Luc02c	0,024	2	2	TRS/CSR/Ex1_2_Luc02c_C
			timeout	TRS/CSR/Ex1_2_Luc02c_GM
			timeout	TRS/CSR/Ex1_2_Luc02c_iGM
TRS/CSR/Ex14_AEGL02	0,074	1	timeout	TRS/CSR/Ex14_AEGL02_C
			1	TRS/CSR/Ex14_AEGL02_GM
			7	TRS/CSR/Ex14_AEGL02_iGM
TRS/CSR/Ex14_Luc06	0,033	11	timeout	TRS/CSR/Ex14_Luc06_C
			11	TRS/CSR/Ex14_Luc06_GM
			timeout	TRS/CSR/Ex14_Luc06_iGM
TRS/CSR/Ex15_Luc06	0,023	0	2	TRS/CSR/Ex15_Luc06_C
			0	TRS/CSR/Ex15_Luc06_GM
			1	TRS/CSR/Ex15_Luc06_iGM
TRS/CSR/Ex15_Luc98	0,035	2	11	TRS/CSR/Ex15_Luc98_C
			2	TRS/CSR/Ex15_Luc98_GM
			timeout	TRS/CSR/Ex15_Luc98_iGM
TRS/CSR/Ex16_Luc06	0,023	0	timeout	TRS/CSR/Ex16_Luc06_C
			0	TRS/CSR/Ex16_Luc06_GM
			11	TRS/CSR/Ex16_Luc06_iGM
TRS/CSR/Ex18_Luc06	0,024	0	3	TRS/CSR/Ex18_Luc06_C
			0	TRS/CSR/Ex18_Luc06_GM
			1	TRS/CSR/Ex18_Luc06_iGM
TRS/CSR/Ex1_GL02a	0,064	1	timeout	TRS/CSR/Ex1_GL02a_C
			1	TRS/CSR/Ex1_GL02a_GM
			2	TRS/CSR/Ex1_GL02a_iGM
TRS/CSR/Ex1_GM03	timeout	timeout	timeout	TRS/CSR/Ex1_GM03_C
			timeout	TRS/CSR/Ex1_GM03_GM
			timeout	TRS/CSR/Ex1_GM03_iGM
TRS/CSR/Ex1_GM99	0,023	0	timeout	TRS/CSR/Ex1_GM99_C
			0	TRS/CSR/Ex1_GM99_GM
			timeout	TRS/CSR/Ex1_GM99_iGM
TRS/CSR/Ex1_Luc02b	0,035	timeout	timeout	TRS/CSR/Ex1_Luc02b_C
			timeout	TRS/CSR/Ex1_Luc02b_GM
			timeout	TRS/CSR/Ex1_Luc02b_iGM
TRS/CSR/Ex1_Luc04b	0,035	2	10	TRS/CSR/Ex1_Luc04b_C
			2	TRS/CSR/Ex1_Luc04b_GM
			18	TRS/CSR/Ex1_Luc04b_iGM
TRS/CSR/Ex1_Zan97	0,052	0	6	TRS/CSR/Ex1_Zan97_C
			0	TRS/CSR/Ex1_Zan97_GM
			0	TRS/CSR/Ex1_Zan97_iGM
TRS/CSR/Ex23_Luc06	0,023	0	11	TRS/CSR/Ex23_Luc06_C
			0	TRS/CSR/Ex23_Luc06_GM
			1	TRS/CSR/Ex23_Luc06_iGM
TRS/CSR/Ex24_GM04	0,024	0	timeout	TRS/CSR/Ex24_GM04_C
			0	TRS/CSR/Ex24_GM04_GM
			1	TRS/CSR/Ex24_GM04_iGM
TRS/CSR/Ex24_Luc06	0,024	0	timeout	TRS/CSR/Ex24_Luc06_C
			0	TRS/CSR/Ex24_Luc06_GM
			10	TRS/CSR/Ex24_Luc06_iGM
TRS/CSR/Ex25_Luc06	0,033	0	12	TRS/CSR/Ex25_Luc06_C
			0	TRS/CSR/Ex25_Luc06_GM
			6	TRS/CSR/Ex25_Luc06_iGM
TRS/CSR/Ex26_Luc03b	0,036	22	timeout	TRS/CSR/Ex26_Luc03b_C
			22	TRS/CSR/Ex26_Luc03b_GM
			timeout	TRS/CSR/Ex26_Luc03b_iGM
TRS/CSR/Ex2_Luc02a	0,039	timeout	timeout	TRS/CSR/Ex2_Luc02a_C
			timeout	TRS/CSR/Ex2_Luc02a_GM
			timeout	TRS/CSR/Ex2_Luc02a_iGM
TRS/CSR/Ex2_Luc03b	0,034	4	9	TRS/CSR/Ex2_Luc03b_C
			4	TRS/CSR/Ex2_Luc03b_GM
			timeout	TRS/CSR/Ex2_Luc03b_iGM
TRS/CSR/Ex3_12_Luc96a	0,035	timeout	timeout	TRS/CSR/Ex3_12_Luc96a_C
			timeout	TRS/CSR/Ex3_12_Luc96a_GM
			timeout	TRS/CSR/Ex3_12_Luc96a_iGM
TRS/CSR/Ex3_2_Luc97	timeout	timeout	timeout	TRS/CSR/Ex3_2_Luc97_C
			timeout	TRS/CSR/Ex3_2_Luc97_GM
			timeout	TRS/CSR/Ex3_2_Luc97_iGM
TRS/CSR/Ex3_3_25_Bor03	0,035	8	16	TRS/CSR/Ex3_3_25_Bor03_C
			8	TRS/CSR/Ex3_3_25_Bor03_GM
			timeout	TRS/CSR/Ex3_3_25_Bor03_iGM
TRS/CSR/Ex4_4_Luc96b	0,033	0	1	TRS/CSR/Ex4_4_Luc96b_C
			0	TRS/CSR/Ex4_4_Luc96b_GM
			0	TRS/CSR/Ex4_4_Luc96b_iGM
TRS/CSR/Ex4_7_15_Bor03	0,053	0	timeout	TRS/CSR/Ex4_7_15_Bor03_C
			0	TRS/CSR/Ex4_7_15_Bor03_GM
			9	TRS/CSR/Ex4_7_15_Bor03_iGM
TRS/CSR/Ex4_7_37_Bor03	0,061	timeout	timeout	TRS/CSR/Ex4_7_37_Bor03_C
			timeout	TRS/CSR/Ex4_7_37_Bor03_GM
			timeout	TRS/CSR/Ex4_7_37_Bor03_iGM
TRS/CSR/Ex4_7_56_Bor03	0,034	timeout	timeout	TRS/CSR/Ex4_7_56_Bor03_C
			timeout	TRS/CSR/Ex4_7_56_Bor03_GM
			timeout	TRS/CSR/Ex4_7_56_Bor03_iGM
TRS/CSR/Ex4_7_77_Bor03	0,023	0	1	TRS/CSR/Ex4_7_77_Bor03_C
			0	TRS/CSR/Ex4_7_77_Bor03_GM
			1	TRS/CSR/Ex4_7_77_Bor03_iGM
TRS/CSR/Ex49_GM04	0,357	3	timeout	TRS/CSR/Ex49_GM04_C
			3	TRS/CSR/Ex49_GM04_GM
			29	TRS/CSR/Ex49_GM04_iGM
TRS/CSR/Ex4_DLMMU04	2,668	timeout	timeout	TRS/CSR/Ex4_DLMMU04_C
			timeout	TRS/CSR/Ex4_DLMMU04_GM
			timeout	TRS/CSR/Ex4_DLMMU04_iGM
TRS/CSR/Ex4_Zan97	0,036	timeout	timeout	TRS/CSR/Ex4_Zan97_C
			timeout	TRS/CSR/Ex4_Zan97_GM
			timeout	TRS/CSR/Ex4_Zan97_iGM
TRS/CSR/Ex5_7_Luc97	timeout	timeout	timeout	TRS/CSR/Ex5_7_Luc97_C
			timeout	TRS/CSR/Ex5_7_Luc97_GM
			timeout	TRS/CSR/Ex5_7_Luc97_iGM
TRS/CSR/Ex5_DLMMU04	0,037	16	timeout	TRS/CSR/Ex5_DLMMU04_C
			16	TRS/CSR/Ex5_DLMMU04_GM
			timeout	TRS/CSR/Ex5_DLMMU04_iGM
TRS/CSR/Ex5_Zan97	0,075	0	3	TRS/CSR/Ex5_Zan97_C
			0	TRS/CSR/Ex5_Zan97_GM
			4	TRS/CSR/Ex5_Zan97_iGM
TRS/CSR/Ex6_15_AEL02	timeout	timeout	timeout	TRS/CSR/Ex6_15_AEL02_C
			timeout	TRS/CSR/Ex6_15_AEL02_GM
			timeout	TRS/CSR/Ex6_15_AEL02_iGM
TRS/CSR/Ex6_9_Luc02c	0,033	4	4	TRS/CSR/Ex6_9_Luc02c_C
			timeout	TRS/CSR/Ex6_9_Luc02c_GM
			timeout	TRS/CSR/Ex6_9_Luc02c_iGM
TRS/CSR/Ex6_GM04	0,024	0	6	TRS/CSR/Ex6_GM04_C
			0	TRS/CSR/Ex6_GM04_GM
			0	TRS/CSR/Ex6_GM04_iGM
TRS/CSR/Ex6_Luc98	0,033	1	4	TRS/CSR/Ex6_Luc98_C
			1	TRS/CSR/Ex6_Luc98_GM
			9	TRS/CSR/Ex6_Luc98_iGM

TRS/CSR/Ex7_BLR02	0,036	timeout	timeout	TRS/CSR/Ex7_BLR02_C
			timeout	TRS/CSR/Ex7_BLR02_GM
			timeout	TRS/CSR/Ex7_BLR02_iGM
TRS/CSR/Ex8_BLR02	0,037	timeout	timeout	TRS/CSR/Ex8_BLR02_C
			timeout	TRS/CSR/Ex8_BLR02_GM
			timeout	TRS/CSR/Ex8_BLR02_iGM
TRS/CSR/Ex9_BLR02	0,036	5	12	TRS/CSR/Ex9_BLR02_C
			5	TRS/CSR/Ex9_BLR02_GM
			41	TRS/CSR/Ex9_BLR02_iGM
TRS/CSR/Ex9_Luc04	0,024	timeout	timeout	TRS/CSR/Ex9_Luc04_C
			timeout	TRS/CSR/Ex9_Luc04_GM
			timeout	TRS/CSR/Ex9_Luc04_iGM
TRS/CSR/Ex9_Luc06	0,024	0	timeout	TRS/CSR/Ex9_Luc06_C
			0	TRS/CSR/Ex9_Luc06_GM
			8	TRS/CSR/Ex9_Luc06_iGM
TRS/CSR/ExAppendixB_AEL03	0,045	timeout	timeout	TRS/CSR/ExAppendixB_AEL03_C
			timeout	TRS/CSR/ExAppendixB_AEL03_GM
			timeout	TRS/CSR/ExAppendixB_AEL03_iGM
TRS/CSR/ExConc_Zan97	0,023	0	2	TRS/CSR/ExConc_Zan97_C
			0	TRS/CSR/ExConc_Zan97_GM
			1	TRS/CSR/ExConc_Zan97_iGM
TRS/CSR/ExIntrod_GM01	0,035	8	16	TRS/CSR/ExIntrod_GM01_C
			8	TRS/CSR/ExIntrod_GM01_GM
			36	TRS/CSR/ExIntrod_GM01_iGM
TRS/CSR/ExIntrod_GM04	0,035	0	14	TRS/CSR/ExIntrod_GM04_C
			0	TRS/CSR/ExIntrod_GM04_GM
			11	TRS/CSR/ExIntrod_GM04_iGM
TRS/CSR/ExIntrod_GM99	0,037	timeout	timeout	TRS/CSR/ExIntrod_GM99_C
			timeout	TRS/CSR/ExIntrod_GM99_GM
			timeout	TRS/CSR/ExIntrod_GM99_iGM
TRS/CSR/ExIntrod_Zan97	timeout	timeout	timeout	TRS/CSR/ExIntrod_Zan97_C
			timeout	TRS/CSR/ExIntrod_Zan97_GM
			timeout	TRS/CSR/ExIntrod_Zan97_iGM
TRS/CSR/ExProp7_Luc06	0,081	1	timeout	TRS/CSR/ExProp7_Luc06_C
			1	TRS/CSR/ExProp7_Luc06_GM
			41	TRS/CSR/ExProp7_Luc06_iGM
TRS/CSR/ExSec11_1_Luc02a	0,041	timeout	timeout	TRS/CSR/ExSec11_1_Luc02a_C
			timeout	TRS/CSR/ExSec11_1_Luc02a_GM
			timeout	TRS/CSR/ExSec11_1_Luc02a_iGM
TRS/CSR/ExSec4_2_DLMMU04	0,04	timeout	timeout	TRS/CSR/ExSec4_2_DLMMU04_C
			timeout	TRS/CSR/ExSec4_2_DLMMU04_GM
			timeout	TRS/CSR/ExSec4_2_DLMMU04_iGM
TRS/CSR/ExSec11_1_Luc02a	0,041	timeout	timeout	TRS/CSR/ExSec11_1_Luc02a_C
			timeout	TRS/CSR/ExSec11_1_Luc02a_GM
			timeout	TRS/CSR/ExSec11_1_Luc02a_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete-noand	1,705	timeout	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete-noand_C
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete-noand_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete-noand_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete	timeout	timeout	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete_C
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_complete_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds-noand	0,405	timeout	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds-noand_C
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds-noand_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds-noand_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds	timeout	timeout	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds_C
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nokinds_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts-noand	timeout	timeout	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts-noand_C
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts-noand_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts-noand_iGM
TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts	timeout	Don't know	timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts_C
			13	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts_GM
			timeout	TRS/CSR_Maude/lazy-nat-list/OvConsOS_nosorts_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete-noand	0,729	timeout	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete-noand_C
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete-noand_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete-noand_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete	30,336	timeout	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete_C
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_complete_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds-noand	0,249	48	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds-noand_C
			48	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds-noand_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds-noand_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds	5,025	7	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds_C
			7	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nokinds_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts-noand	timeout	Don't know	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts-noand_C
			10	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts-noand_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts-noand_iGM
TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts	timeout	Don't know	timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts_C
			2	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts_GM
			timeout	TRS/CSR_Maude/length-lazy-list/LengthOfFiniteLists_nosorts_iGM
TRS/CSR_Maude/my-nat/MYNAT_complete-noand	0,333	timeout	timeout	TRS/CSR_Maude/my-nat/MYNAT_complete-noand_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_complete-noand_GM
			timeout	TRS/CSR_Maude/my-nat/MYNAT_complete-noand_iGM
TRS/CSR_Maude/my-nat/MYNAT_complete	22,267	timeout	timeout	TRS/CSR_Maude/my-nat/MYNAT_complete_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_complete_GM
			timeout	TRS/CSR_Maude/my-nat/MYNAT_complete_iGM
TRS/CSR_Maude/my-nat/MYNAT_nokinds-noand	0,054	timeout	timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds-noand_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds-noand_GM
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds-noand_iGM
TRS/CSR_Maude/my-nat/MYNAT_nokinds	2,619	timeout	timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds_GM
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nokinds_iGM
TRS/CSR_Maude/my-nat/MYNAT_nosorts-noand	0,039	timeout	timeout	TRS/CSR_Maude/my-nat/MYNAT_nosorts-noand_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nosorts-noand_GM
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nosorts-noand_iGM
TRS/CSR_Maude/my-nat/MYNAT_nosorts	0,036	10	10	TRS/CSR_Maude/my-nat/MYNAT_nosorts_C
			timeout	TRS/CSR_Maude/my-nat/MYNAT_nosorts_GM
			18	TRS/CSR_Maude/my-nat/MYNAT_nosorts_iGM
TRS/CSR_Maude/palindrome/PALINDROME_complete-noand	0,32	timeout	timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete-noand_C
			timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete-noand_GM
			timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete-noand_iGM
TRS/CSR_Maude/palindrome/PALINDROME_complete	0,377	timeout	timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete_C
			timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete_GM
			timeout	TRS/CSR_Maude/palindrome/PALINDROME_complete_iGM
TRS/CSR_Maude/palindrome/PALINDROME_nokinds-noand	0,054	7	timeout	TRS/CSR_Maude/palindrome/PALINDROME_nokinds-noand_C
			7	TRS/CSR_Maude/palindrome/PALINDROME_nokinds-noand_GM
			timeout	TRS/CSR_Maude/palindrome/PALINDROME_nokinds-noand_iGM
TRS/CSR_Maude/palindrome/PALINDROME_nokinds	0,08	2	timeout	TRS/CSR_Maude/palindrome/PALINDROME_nokinds_C
			2	TRS/CSR_Maude/palindrome/PALINDROME_nokinds_GM
			52	TRS/CSR_Maude/palindrome/PALINDROME_nokinds_iGM
TRS/CSR_Maude/palindrome/PALINDROME_nosorts-noand	0,036	1	11	TRS/CSR_Maude/palindrome/PALINDROME_nosorts-noand_C
			1	TRS/CSR_Maude/palindrome/PALINDROME_nosorts-noand_GM
			7	TRS/CSR_Maude/palindrome/PALINDROME_nosorts-noand_iGM
TRS/CSR_Maude/palindrome/PALINDROME_nosorts	0,035	0	5	TRS/CSR_Maude/palindrome/PALINDROME_nosorts_C
			0	TRS/CSR_Maude/palindrome/PALINDROME_nosorts_GM
			6	TRS/CSR_Maude/palindrome/PALINDROME_nosorts_iGM

TRS/CSR_Maude/peanoSimple/MYNAT_complete-noand	0,131	timeout	timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete-noand_C
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete-noand_GM
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete-noand_iGM
TRS/CSR_Maude/peanoSimple/MYNAT_complete	0,24	timeout	timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete_C
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete_GM
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_complete_iGM
TRS/CSR_Maude/peanoSimple/MYNAT_nokinds-noand	0,043	22	timeout	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds-noand_C
			22	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds-noand_GM
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds-noand_iGM
TRS/CSR_Maude/peanoSimple/MYNAT_nokinds	0,066	31	timeout	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds_C
			31	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds_GM
			timeout	TRS/CSR_Maude/peanoSimple/MYNAT_nokinds_iGM
TRS/CSR_Maude/peanoSimple/MYNAT_nosorts-noand	0,036	1	timeout	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts-noand_C
			1	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts-noand_GM
			19	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts-noand_iGM
TRS/CSR_Maude/peanoSimple/MYNAT_nosorts	0,033	0	7	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts_C
			0	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts_GM
			5	TRS/CSR_Maude/peanoSimple/MYNAT_nosorts_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete-noand	4,04	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete-noand_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete-noand_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete-noand_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete	timeout	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_complete_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds-noand	0,62	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds-noand_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds-noand_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds-noand_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds	timeout	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nokinds_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts-noand	0,055	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts-noand_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts-noand_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts-noand_iGM
TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts	0,041	timeout	timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts_C
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts_GM
			timeout	TRS/CSR_Maude/PEPM04/LISTUTILITIES_nosorts_iGM
TRS/aprove08-csr/csrdiv	0,297	timeout	timeout	TRS/aprove08-csr/csrdiv_C
			timeout	TRS/aprove08-csr/csrdiv_GM
			timeout	TRS/aprove08-csr/csrdiv_iGM
TRS/aprove08-csr/emmes	timeout	timeout	timeout	TRS/aprove08-csr/emmes_C
			timeout	TRS/aprove08-csr/emmes_GM
			timeout	TRS/aprove08-csr/emmes_iGM
TRS/endrullis08/cariboo_ex1	0,036	-	-	TRS/endrullis08/cariboo_ex1_C
			-	TRS/endrullis08/cariboo_ex1_GM
			-	TRS/endrullis08/cariboo_ex1_iGM
TRS/endrullis08/cariboo_ex2	0,207	-	timeout	TRS/endrullis08/cariboo_ex2_C
			-	TRS/endrullis08/cariboo_ex2_GM
			-	TRS/endrullis08/cariboo_ex2_iGM
TRS/endrullis08/cariboo_ex3	0,056	-	-	TRS/endrullis08/cariboo_ex3_C
			-	TRS/endrullis08/cariboo_ex3_GM
			-	TRS/endrullis08/cariboo_ex3_iGM
TRS/endrullis08/cariboo_ex4	0,053	-	-	TRS/endrullis08/cariboo_ex4_C
			-	TRS/endrullis08/cariboo_ex4_GM
			-	TRS/endrullis08/cariboo_ex4_iGM
TRS/endrullis08/cariboo_ex5	0,073	-	-	TRS/endrullis08/cariboo_ex5_C
			-	TRS/endrullis08/cariboo_ex5_GM
			-	TRS/endrullis08/cariboo_ex5_iGM
TRS/endrullis08/cariboo_ex6	0,07	-	timeout	TRS/endrullis08/cariboo_ex6_C
			-	TRS/endrullis08/cariboo_ex6_GM
			-	TRS/endrullis08/cariboo_ex6_iGM
TRS/endrullis08/ex5.3	0,036	-	-	TRS/endrullis08/ex5.3_C
			-	TRS/endrullis08/ex5.3_GM
			-	TRS/endrullis08/ex5.3_iGM
TRS/endrullis08/ex5.4	0,223	-	-	TRS/endrullis08/ex5.4_C
			-	TRS/endrullis08/ex5.4_GM
			-	TRS/endrullis08/ex5.4_iGM
TRS/endrullis08/ex5.5	0,143	-	timeout	TRS/endrullis08/ex5.5_C
			-	TRS/endrullis08/ex5.5_GM
			timeout	TRS/endrullis08/ex5.5_iGM
TRS/endrullis08/ex5.6	0,056	-	timeout	TRS/endrullis08/ex5.6_C
			-	TRS/endrullis08/ex5.6_GM
			-	TRS/endrullis08/ex5.6_iGM
TRS/endrullis08/ex5.7	0,078	-	-	TRS/endrullis08/ex5.7_C
			-	TRS/endrullis08/ex5.7_GM
			-	TRS/endrullis08/ex5.7_iGM
TRS/endrullis08/ex5.8	0,037	-	-	TRS/endrullis08/ex5.8_C
			-	TRS/endrullis08/ex5.8_GM
			-	TRS/endrullis08/ex5.8_iGM
TRS/endrullis08/f20	3,762	timeout	timeout	TRS/endrullis08/f20_C
			timeout	TRS/endrullis08/f20_GM
			timeout	TRS/endrullis08/f20_iGM
TRS/endrullis08/f30	0,232	-	timeout	TRS/endrullis08/f30_C
			-	TRS/endrullis08/f30_GM
			timeout	TRS/endrullis08/f30_iGM
TRS/endrullis08/f40.trs	0,447	-	timeout	TRS/endrullis08/f40_C
			-	TRS/endrullis08/f40_GM
			timeout	TRS/endrullis08/f40_iGM
TRS/endrullis08/f4	0,171	-	-	TRS/endrullis08/f4_C
			-	TRS/endrullis08/f4_GM
			-	TRS/endrullis08/f4_iGM
TRS/endrullis08/morse	4,447	-	timeout	TRS/endrullis08/morse_C
			-	TRS/endrullis08/morse_GM
			timeout	TRS/endrullis08/morse_iGM
YES	**95/109**	**60/109**	**132/327**	
YES AVERAGE TIME (s)	0,773798165	1,532110092		
SUCESS PERCENTAGE	87,20%	55,00%		

20

Appendix B: Detailed Benchmarks on $A \vee C$

	Features
Processor	Intel(R) Core(TM)2 Duo 2.16GHz
RAM	1GB SDRAM
OS	Ubuntu-Linux 2.6.24-19-generic
Program Versions	MuTerm + SMT Solver
Database	TPDB 7.0
Timeout	60s (300s)
Date	10.15.10

Examples	muterm	AproVE
TRS/AProVE_AC_04/AC01.trs	0.27	1.421
TRS/AProVE_AC_04/AC02.trs	0.269	1.418
TRS/AProVE_AC_04/AC03.trs	1.471	1.532
TRS/AProVE_AC_04/AC04.trs	1.56	3.415
TRS/AProVE_AC_04/AC05.trs	1.83	3.824
TRS/AProVE_AC_04/AC06.trs	4.161	1.477
TRS/AProVE_AC_04/AC07.trs	0.929	2.895
TRS/AProVE_AC_04/AC09.trs	9.291	1.482
TRS/AProVE_AC_04/AC10.trs	5.356	1.485
TRS/AProVE_AC_04/AC11.trs	0.281	1.658
TRS/AProVE_AC_04/AC12.trs	0.638	1.531
TRS/AProVE_AC_04/AC13.trs	8.555	1.49
TRS/AProVE_AC_04/AC14.trs	5.446	1.489
TRS/AProVE_AC_04/AC15.trs	0.369	1.473
TRS/AProVE_AC_04/AC16.trs	1.046	1.589
TRS/AProVE_AC_04/AC17.trs	7.251	2.333
TRS/AProVE_AC_04/AC18.trs	9.049	16.146
TRS/AProVE_AC_04/AC19.trs	0.266	1.528
TRS/AProVE_AC_04/AC20.trs	1.538	3.459
TRS/AProVE_AC_04/AC21.trs	10.386	13.311
TRS/AProVE_AC_04/AC22.trs	54.007	14.057
TRS/AProVE_AC_04/AC23.trs	17.928	2.682
TRS/AProVE_AC_04/AC24.trs	2.611	2.236
TRS/AProVE_AC_04/AC26.trs	5.773	1.536
TRS/AProVE_AC_04/AC27.trs	23.193	1.546
TRS/AProVE_AC_04/AC28.trs	52.257	10.312
TRS/AProVE_AC_04/AC41.trs	0.431	2.468
TRS/AProVE_AC_04/AC48.trs	1.739	1.483
TRS/AProVE_AC_04/AC49.trs	0.574	3.687
TRS/AProVE_AC_04/AC50.trs	0.774	5.216
TRS/AProVE_AC_04/AC51.trs	0.634	1.897
TRS/AProVE_AC_04/AC52.trs	2.222	4.521
TRS/AProVE_AC_04/AC53.trs	0.165	2.966
TRS/AProVE_AC_04/AC54.trs	0.717	4.058
TRS/AProVE_AC_04/IJCAR_AC1.trs	2.188	3.859
TRS/Mixed_AC_and_C/AC08.trs	3.998	4.144
TRS/Mixed_AC_and_C/AC29.trs	0.433	2.815
TRS/Mixed_AC_and_C/AC47.trs	0.707	5.448
TRS/Mixed_AC_and_C/rationals.trs	60	60
TRS/Mixed_AC/BAG_complete-noand.trs	23.91	60
TRS/Mixed_AC/BAG_complete.trs	22.867	60
TRS/Mixed_AC/BAG_nokinds-noand.trs	6.312	13.33
TRS/Mixed_AC/BAG_nokinds.trs	4.74	15.551

TRS/Mixed_AC/BAG_nosorts-noand.trs	2.795	3.623
TRS/Mixed_AC/BAG_nosorts.trs	3.234	3.019
TRS/Mixed_AC/bag-sum-prod-bin.trs	3.779	2.226
TRS/Mixed_AC/bag-sum-prod-distr.trs	22.476	2.6
TRS/Mixed_AC/bag-sum-prod.trs	2.265	1.536
TRS/Mixed_AC/boolean_rings.trs	4.76	1.528
TRS/Mixed_AC/differ.trs	0.625	5.445
TRS/Mixed_AC/intersect.trs	4.394	2.138
TRS/Mixed_AC/kusakari1.trs	0.328	6.055
TRS/Mixed_AC/RENAMED-BOOL_complete-noand.trs	60	60
TRS/Mixed_AC/RENAMED-BOOL_complete.trs	60	60
TRS/Mixed_AC/RENAMED-BOOL_nokinds-noand.trs	24.786	14.194
TRS/Mixed_AC/RENAMED-BOOL_nokinds.trs	25.811	35.421
TRS/Mixed_AC/RENAMED-BOOL_nosorts-noand.trs	7.696	15.986
TRS/Mixed_AC/RENAMED-BOOL_nosorts.trs	60	5.115
TRS/Mixed_AC/sequent_modulo.trs	155.348	30.213
TRS/Mixed_C/AC42.trs	0.36	2.239
TRS/Mixed_C/AC43.trs	0.327	2.389
TRS/Mixed_C/AC44.trs	0.412	2.564
TRS/Mixed_C/AC45.trs	0.369	2.507
TRS/Mixed_C/AC46.trs	0.993	3.921
TRS/Mixed_C/maude2.trs	2.257	4.327
TRS/Mixed_C/PEANO-NAT_complete-noand.trs	60	60
TRS/Mixed_C/PEANO-NAT_complete.trs	60	60
TRS/Mixed_C/PEANO-NAT_nokinds-noand.trs	60	17.583
TRS/Mixed_C/PEANO-NAT_nokinds.trs	41.51	16.114
TRS/Mixed_C/PEANO-NAT_nosorts-noand.trs	3.712	7.601
TRS/Mixed_C/PEANO-NAT_nosorts.trs	2.388	7.235
	muterm	AProVE
#TO	8	7
#YES	59	57
#MAYBE	5	5
#NO	0	2
#ERROR	0	0
Avg.YES	6.83	5.123
Avg.NO	0	10.5505
Max. Time	60 (300)	60

21

Appendix C: Detailed Benchmarks on Transformed Maude Examples

	Features
Processor	Intel(R) Core(TM)2 Duo 2.16GHz
RAM	1GB SDRAM
OS	Ubuntu-Linux 2.6.24-19-generic
Program Versions	MuTerm + MultiSolver
Database	MTT examples
Timeout	60s
Date	05.02.11

Examples	muterm	muterm-old
TRS/Maude_AC/blackboard-C-A-B-no-sorts-off-off.trs	1.103	AC not supported
TRS/Maude_AC/blackboard-OS-T-A-B-no-sorts-off-off.trs	1.297	AC not supported
TRS/Maude_AC/data-agents-C-A-B-no-sorts-off-off.trs	39.025	AC not supported
TRS/Maude_AC/die-hard-C-A-B-no-sorts-off-off.trs	1.208	AC not supported
TRS/Maude_AC/die-hard-OS-T-A-B-no-sorts-off-off.trs	10.651	AC not supported
TRS/Maude_AC/die-hard-OS-T-B-O-L--off-off.trs	60	AC not supported
TRS/Maude_AC/dining-philosophers-5-C-A-B-no-sorts-off-off.trs	2.223	AC not supported
TRS/Maude_AC/inf-C-A-B-no-kinds-off-off.trs	60	60
TRS/Maude_AC/inf-C-OS-A-B-no-kinds-off-off.trs	60	60
TRS/Maude_AC/inf-C-OS-B-O-L--off-off.trs	26.424	26.126
TRS/Maude_AC/josephus-C-A-B-no-sorts-off-off.trs	0.08	0.076
TRS/Maude_AC/josephus-generalized-C-A-B-no-sorts-off-off.trs	1.258	AC not supported
TRS/Maude_AC/josephus-generalized-C-OS-B-A-no-sorts-off-off.trs	0.408	AC not supported
TRS/Maude_AC/josephus-OS-T-B-A-no-sorts-off-off.trs	0.085	0.081
TRS/Maude_AC/lazy-list-utilities-C-A-B-no-sorts-off-on.trs	0.041	0.041
TRS/Maude_AC/lazy-list-utilities-C-OS-B-O-L-off-on.trs	0.049	0.049
TRS/Maude_AC/lazy-nat-list-C-OS-B-O-L--off-on.trs	60	60
TRS/Maude_AC/mtt-bool-C-A-B-no-sorts-off-off.trs	0.023	0.023
TRS/Maude_AC/mtt-bool-C-OS-B-A-no-sorts-off-off.trs	0.023	0.024
TRS/Maude_AC/mtt-list-C-A-B-no-kinds-off-off.trs	0.041	0.041
TRS/Maude_AC/mtt-list-C-A-B-no-sorts-off-off.trs	0.033	0.033
TRS/Maude_AC/mtt-map-C-A-B-no-sorts-off-off.trs	0.783	AC not supported
TRS/Maude_AC/mtt-map-C-OS-B-A-no-sorts-off-off.trs	1.055	AC not supported
TRS/Maude_AC/mtt-nat-C-A-B-no-sorts-off-off.trs	0.043	0.043
TRS/Maude_AC/mtt-nat-C-OS-A-B-no-sorts-off-off.trs	0.045	0.045
TRS/Maude_AC/rabbit-hop-C-A-B-no-sorts-off-off.trs	0.122	0.114
TRS/Maude_AC/rabbit-hop-OS-T-A-B-no-sorts-off-off.trs	0.123	0.119
	muterm	muterm-old
Time	86.143	26.84
#TO	4	3
#YES	20	12
#MAYBE	3	9
#NO	0	0
#ERROR	0	0
Avg.YES	2.39	1.72
Max. Time	60	60