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

Ph.D. Thesis

Termination of Narrowing: Automated Proofs and Modularity Properties

Candidate:
José Iborra

Supervisors:
María Alpuente
Santiago Escobar

October 26, 2010

This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02-02, Integrated Action HA 2006-0007, and Generalitat Valenciana GVPRE/2008/113.

Author's address:

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

Abstract

In 1936, Alan Turing proved that *the halting problem*, that is, deciding whether a program terminates, is an undecidable problem for most practical programming languages. Even so, termination is so relevant that a vast number of techniques for proving the termination of programs have been researched in the recent decades. Term rewriting systems provide an abstract theoretical framework ideally suited for the study of termination. In term rewriting systems, evaluation of a term corresponds to the left-to-right, non-deterministic application of rewriting rules.

Narrowing is a generalization of term rewriting that provides a mechanism for automated reasoning. For instance, given a set of rules defining addition and multiplication with natural numbers, rewriting allows to perform *evaluation* of arithmetic expressions, whereas narrowing allows to *solve* equations with variables over arithmetic expressions. This thesis constitutes an in-depth study of the termination properties of narrowing. The contributions are as follows.

First, we identify *classes of systems* where narrowing behaves *well*, in the sense that it always terminates for any system belonging to these classes. Many methods of analysis, such as the semantics-based analysis of program properties by means of narrowing, benefit from this characterization.

Second, we develop an automatic method to prove termination of narrowing *for a given system*, based on the abstract framework of dependency pairs. For the first time, such an automatic method is not restricted to specific classes of TRSs and hence is universally applicable.

Third, we propose another automatic method to prove termination of narrowing *from a given term*, also on top of the framework of dependency pairs. Termination from a given term is relevant for many applications, including the termination analysis of programming languages. Our method generalizes the current state-of-the-art, enabling the study of termination of logic programs in terms of the termination of narrowing, something which was not possible previously.

Fourth and last, the modularity properties of the termination of narrowing are considered in detail. That is, given two systems A and B where narrowing is terminating, determine whether it is still terminating in the union system. This notion of modularity has implications in many of the applications of narrowing. We develop the case of combining systems for equational unification.

Furthermore, the automated approaches of the second and third contributions above have been implemented in NARRADAR, our tool for automatic proofs of termination of narrowing.

Resumen

En 1936 Alan Turing demostró que el *halting problem*, esto es, el problema de decidir si un programa termina o no, es un problema indecidible para la inmensa mayoría de los lenguajes de programación. A pesar de ello, la terminación es un problema tan relevante que en las últimas décadas un gran número de técnicas han sido desarrolladas para demostrar la terminación de forma automática de la máxima cantidad posible de programas. Los sistemas de reescritura de términos proporcionan un marco teórico abstracto perfecto para el estudio de la terminación de programas. En este marco, la evaluación de un término consiste en la aplicación no determinista de un conjunto de reglas de reescritura.

El estrechamiento (*narrowing*) de términos es una generalización de la reescritura que proporciona un mecanismo de razonamiento automático. Por ejemplo, dado un conjunto de reglas que definan la suma y la multiplicación, la reescritura permite *calcular* expresiones aritméticas, mientras que el estrechamiento permite *resolver* ecuaciones con variables. Esta tesis constituye el primer estudio en profundidad de las propiedades de terminación del estrechamiento. Las contribuciones son las siguientes.

En primer lugar, se identifican clases de sistemas en las que el estrechamiento tiene un comportamiento *bueno*, en el sentido de que siempre termina. Muchos métodos de razonamiento automático, como el análisis de la semántica de lenguajes de programación mediante operadores de punto fijo, se benefician de esta caracterización.

En segundo lugar, se introduce un método automático, basado en el marco teórico de *pares de dependencia*, para demostrar la terminación del estrechamiento *en un sistema particular*. Nuestro método es, por primera vez, aplicable a *cualquier* clase de sistemas.

En tercer lugar, se propone un nuevo método para estudiar la terminación del estrechamiento *desde un término particular*, permitiendo el análisis de la terminación de lenguajes de programación. El nuevo método generaliza los métodos existentes de manera fundamental, gracias a lo cual se puede estudiar la terminación de lenguajes de programación lógicos a través de la terminación del estrechamiento, algo que hasta ahora no era posible con los métodos existentes.

En cuarto lugar, se analiza detalladamente la modularidad de la terminación del estrechamiento. Esto es, si los sistemas A y B son terminantes, en qué casos el sistema unión es terminante. La modularidad tiene implicaciones directas en muchas de las aplicaciones del estrechamiento. En concreto desarrollamos el caso de la resolución de ecuaciones simbólicas cuando se combinan varios sistemas ecuacionales.

Además, las técnicas automáticas de la segunda y tercera contribución han sido implementadas en nuestra herramienta de demostración de la terminación del estrechamiento, NARRADAR.

Resum

En 1936 Alan Turing va demostrar que el *halting problem*, és a dir, el problema de decidir si un programa acaba o no, és un problema indecidible per a la immensa majoria dels llenguatges de programació. Tot i això, la terminació és un problema tan rellevant que en les últimes dècades s'ha desenvolupat una gran nombre de técniques per a demostrar la terminació de la màxima quantitat de programes possible de manera automàtica. Els sistemes de reescriptura de termes proporcionen un marc teòric abstracte perfecte per a la caracterizació de les propietats de terminació de programes. En aquest marc, l'avaluació d'un terme consisteix en l'aplicació no determinista d'un conjunt de regles de reescriptura.

L'estretiment (*narrowing*) de termes és una generalització de la reescriptura que proporciona un mecanisme de raonament automàtic. Per exemple, donat un conjunt de regles que definisquen la suma i la multiplicació dels naturals, la reescriptura permet *calcular* expressions aritmètiques, mentre que l'estretiment permet *resoldre* equacions amb variables. Aquesta tesi constitueix el primer estudi en profunditat de les propietats de terminació de l'estretiment. Les contribucions són les següents.

En primer lloc, s'identifiquen classes de sistemes en què l'estretiment té un comportament *bo*, en el sentit que sempre acaba. Molts mètodes de raonament automàtic, como l'anàlisi de les propietats de programes basat en una semàntica computada per mitjà de narrowing, es beneficien d'aquesta caracterització.

En segon lloc, s'introdueix un mètode automàtic, basat en el marc teòric de *parells de dependència*, per a demostrar la terminació de l'estretiment *en un sistema particular*. El nostre mètode es, por primera volta, aplicable a qualsevol classe de sistemes.

En tercer lloc, es proposa un nou mètode per l'estudi de la terminació de l'estretiment *des d'un terme particular*, cosa que permet l'anàlisi de la terminació de programes. El nostre mètode generalitza els mètodes existents de manera fonamental. Gràcies a això, es pot estudiar la terminació de llenguatges de programació lògics a través de la terminació de l'estretiment, un fet que fins ara no era possible amb els mètodes que hi havia.

En quart lloc, s'analitza detalladament la modularitat de la terminació de l'estretiment. Això és, si els sistemes A i B són terminants, en què casos el sistema unió és terminant. La modularitat té implicacions directes en moltes de les aplicacions de l'estretiment. En concret, desenvolupem el cas de la resolució de equacions simbòliques quan es combinen diversos sistemes equacionals.

A més, les tècniques automàtiques de la segona i tercera contribució han sigut implementades en la nostra eina de demostració de la terminació de l'estretiment, NARRADAR.

Acknowledgements

First of all, I would like to thank María Alpuente for her help, guidance and patience during these years. It was a pleasure to work with her. I am also equally thankful to Santiago Escobar; from him I finally learnt the foundations of research during long blackboard sessions in the summer of 2007. Most of the work in this thesis is directly or indirectly due to them. Gracias María ! Gracias Santi !

During these years I was fortunate to have German Vidal at only two doors distance. He has devoted so much time to helping me when I struggled, that probably this thesis wouldn't exist if he hadn't been there.

I was also privileged to be sharing the laboratory with Raúl Gutierrez. How many times has he helped with a technical question. have we sketched ideas over a blackboard, shared interesting references, or designed algorithms for our implementations on the way to lunch. He also kindly proofreaded parts of this thesis. In sum, this PhD would have been a lot less fun without him.

I want to thank Alicia Villanueva, Marco Feliu, Beatriz Alarcón, Josep Silva, Salvador Tamarit, Rafa Navarro, César Ferri, Sonia Santiago, Daniel Romero, Mauricio Alba, and the rest of the ELP team, for making the lab a great place to work, and for enduring my long rants about functional programming.

Part of the work of thesis was developed during research stays. I want to thank Moreno Falaschi and specially Marco Comini and Demis Ballis for being such amazing hosts and making my visits to Italy a highlight of my recent years. I am also deeply indebted to Peter Schneider-Kamp, for allowing me to stay (twice) in his home at Odense, and abuse his time and knowledge. He taught me many practical lessons about research that I have found useful in the late part of this PhD. I also found that the beach in Denmark has nothing to do with the beach as we know it in Spain.

During this thesis I particularly enjoyed the implementation aspects, which have given me the opportunity to learn and use functional programming. Thanks to Google I had the opportunity in the summer of 2006 to collaborate more deeply with the Haskell community, which has been a source of inspiration during these years. Thanks are in order to my partner in crime during that project, Bernie Pope, who has supported me in numerous occasions since then.

I am indebted to Salvador Lucas, who spotted an email I had sent to the Haskell mailing list and offered me to start a collaboration which turned into a PhD. I want to thank Oscar Pastor and Vicente Pelechano for their understanding after I presented them my resignation in 2006, and my great colleagues of Lab. 204, Gonzalo, Vicky, Paco, Marta, and the rest of the people in the OO-Method group who I worked with during those early months.

This thesis wouldn't have been possible without the financial support of Universidad Politécnica de Valencia and Ministerio de Educación y Ciencia.

Finally, I owe a big thanks to my mother, for her unconditional support, and to my father, who bought me a computer back in 1992 and taught me about programming. He has been and will always be the example to follow.

And to the muse, who visits on Sunday nights.

Contents

List of Definitions

1 Introduction

Narrowing [*Fay*, 1979] is a generalization of term rewriting that allows free variables in terms (as in logic programming) and replaces pattern matching by syntactic unification so that it subsumes both rewriting and SLD-resolution [*Hanus*, 1994]. Narrowing has many important applications including:

- **Execution of functional–logic programming languages and logic languages [*Dershowitz*, 1995; *Hanus*, 1994; *Meseguer*, 1992].** Narrowing is the operational mechanism of functional-logic languages such as Curry [*Hanus*, 1994], or Toy [*López-Fraguas and Sánchez-Hernández*, 1999]. Moreover, a restriction of narrowing called *basic* narrowing (cf. Chapter 3) is fundamentally equivalent to SLD resolution [*Bosco et al.*, 1988], the operational mechanism of logic languages such as Prolog.

- **Equational unification [*Hullot*, 1980].** Equational unification is a general procedure for finding solutions of symbolic equations. Due to non-termination, narrowing behaves as a semi-decision procedure for the problem of equational unification in a wide variety of equational theories.

- **Symbolic reachability [*Meseguer and Thati*, 2007]** that is, given a set of rules $\mathcal{R}$ and initial and final terms s and t, narrowing is able to find the "most general" solution σ for the variables of s and t, such that $s\sigma$ rewrites to $t\sigma$ in a finite number of steps,

- **Automated proofs of termination of term rewriting [*Giesl et al.*, 2006b; *Arts and Zantema*, 1996]**. Narrowing is used in several techniques for the approximation and refinement of so-called *dependency graphs*,

- **Automated proofs of non-termination in term rewriting** Narrowing is used as a mechanism for *unfolding* rewriting rules in proofs of non-termination of term rewriting systems [*Payet*, 2006, 2008; *Giesl et al.*, 2005b].

- **Verification of security policies [*Kirchner et al.*, 2008] and cryptographic protocols [*Escobar et al.*, 2009]**. Narrowing is used to perform automated reasoning over suitable definitions, by means of rewriting rules, of security policies and protocols, in order to establish the satisfaction of relevant properties.

- **Type checking** [***Sheard*, 2006**], where narrowing can replace syntactic unification in Damas-Milner [*Damas and Milner*, 1982] based type systems extended to dependent types.

- **Equational constraint solving** [***Alpuente et al.*, 1993, 1995a**]. Narrowing provides a complete semantic unification procedure for the equational constraint logic programming scheme that parameterizes the CLP framework with respect to a Horn equational theory.

- **Symbolic model checking of infinite** [***Escobar and Meseguer*, 2007**] **state systems**. Narrowing is a complete symbolic method for model checking reachability properties of systems expressed by means of rewriting rules. Narrowing can also be used to model check systems with state predicates, which correspond to Kripke structures on which ACTL* and LTL formulas can be verified.

- **Semantics of term rewriting systems** [***Alpuente et al.*, 2010a**] In joint work [*Alpuente et al.*, 2010a], we propose an immediate consequences operator which relies on narrowing to compute the compact semantics of a wide class of term rewriting systems.

Termination of narrowing itself is of great interest to these applications. Without it, many of these applications are simply not possible or their usefulness is seriously affected. For instance, for all the applications related with reasoning, including equational unification, verification, and reachability, without a termination proof narrowing can only provide a semi-decision procedure.

Example 1.1. *Consider the following term rewriting system (TRS) defining the addition* add *on natural numbers built from* 0 *and* s[1] *:*

$$\mathtt{add}(\mathtt{0},y) \to y \tag{R1}$$
$$\mathtt{add}(\mathtt{s}(x),y) \to \mathtt{s}(\mathtt{add}(x,y)) \tag{R2}$$

Narrowing allows us to prove that the formula

$$\exists w \exists z \ s.t.\ \mathtt{add}(w,\mathtt{s}(\mathtt{0})) = \mathtt{s}(\mathtt{s}(z))$$

holds by computing the solution

$$\{w \mapsto \mathtt{s}(\mathtt{0}),\ z \mapsto \mathtt{0}\}$$

whereas it cannot prove that the formula

$$\exists w \ s.t.\ \mathtt{add}(w,\mathtt{s}(\mathtt{0})) = \mathtt{0}$$

[1]Along this thesis, variables are written in italic font and function symbols are in typewriter font.

does not hold. There are infinitely many narrowing derivations issuing from the input expression $\mathtt{add}(w, \mathtt{s}(0))$*:*

$$\begin{array}{ll}
\underline{\mathtt{add}(w,\mathtt{s}(0))} & \leadsto_{\{w\mapsto 0\},(R1)} \mathtt{s}(0) \\
\underline{\mathtt{add}(w,\mathtt{s}(0))} & \leadsto_{\{w\mapsto \mathtt{s}(x)\},(R2)} \mathtt{s}(\underline{\mathtt{add}(x,\mathtt{s}(0))}) \leadsto_{\{x\mapsto 0\},(R1)} \mathtt{s}(\mathtt{s}(0)) \\
\underline{\mathtt{add}(w,\mathtt{s}(0))} & \leadsto_{\{w\mapsto \mathtt{s}(x)\},(R2)} \mathtt{s}(\underline{\mathtt{add}(x,\mathtt{s}(0))}) \leadsto_{\{x\mapsto \mathtt{s}(x')\},(R2)} \mathtt{s}(\mathtt{s}(\underline{\mathtt{add}(x',\mathtt{s}(0))})) \\
 & \leadsto_{\{x\mapsto 0\},(R1)} \mathtt{s}(\mathtt{s}(\mathtt{s}(0))) \\
\vdots &
\end{array}$$

(where at each step, the narrowing relation $\leadsto$ *is labelled with the rule used and the substitution computed*[2]*, and the reduced subterm is underlined).*

The following infinite narrowing derivation resulting from applying rule (R2) infinitely many times can also be proven:

$$\underline{\mathtt{add}(w,\mathtt{s}(0))} \leadsto_{\{w\mapsto \mathtt{s}(x)\},(R2)} \mathtt{s}(\underline{\mathtt{add}(x,\mathtt{s}(0))}) \leadsto_{\{x\mapsto \mathtt{s}(x')\},(R2)} \mathtt{s}(\mathtt{s}(\underline{\mathtt{add}(x',\mathtt{s}(0))}))\cdots$$

Termination of narrowing has implications on many of the applications mentioned above. Briefly, some of these are:

Proofs of termination of functional-logic and logic programs. As narrowing is the operational mechanism of functional-logic languages, and it is closely related to the operational mechanism of logic programs, termination of narrowing can be used to prove termination of such programs.

Proofs of termination of imperative programs. Semantic-based techniques can prove termination of imperative programs in terms of termination of logic programs [*Albert et al.*, 2008], which in turn can be proven in terms of termination of narrowing.

Improved proofs of termination and non-termination of term rewriting. As mentioned before, narrowing plays a role in several techniques for proofs of termination and non-termination of term rewriting. Often, some assumptions and/or requirements are needed in order to ensure the termination of narrowing. The results in this thesis might enable some of those restrictions to be relaxed or dropped where appropriate.

Decidable equational unification. Termination of narrowing, combined with completeness, ensures that narrowing is a decidable procedure for the problem of equational unification [*Alpuente et al.*, 2010b].

Decidable of symbolic reachability. Again, termination combined with completeness provides a decidable procedure for solving problems of symbolic reachability.

Verification of security policies and cryptographic protocols. Again due to decidability, termination of narrowing is essential for enabling the checking of most relevant properties.

[2] Restricted to the input variables.

Semantics of term rewriting systems. The results in this thesis regarding classes of systems where narrowing terminates enable us to establish the completeness of an immediate consequences operator which uses narrowing to compute a compact semantics for term rewriting systems [*Alpuente et al.*, 2010a].

1.1 State of the Art

Termination of term rewriting systems has been studied extensively in the recent decades, and it can safely be said that it has become a hot topic in the top conferences and journals of the area. We refrain from giving an enumeration of the endless references about the topic and refer the reader to [*Ohlebusch*, 2002] for a survey on the topic.

On the other hand, there are few works in the literature which focus on the study of the termination of narrowing. Existing termination results for narrowing have been obtained as a by-product of other works that address the decidability of equational unification or the completeness of narrowing-based equational unification algorithms. To facilitate the understanding of our results, let us first summarize the existing completeness results for narrowing as a procedure to solve equational unification as well as reachability goals.

1.1.1 Existing completeness results for narrowing

[*Fay*, 1979] and [*Hullot*, 1980] demonstrated that narrowing is a complete method for solving equational unification goals $s_1 = t_1, \ldots, s_n = t_n$ in an equational theory defined by a canonical term rewriting system $\mathcal{R}$. In the equational setting, completeness means that, for every solution ρ to a given equational goal G (i.e., $\mathcal{R} \vdash s_i\rho = t_i\rho$, for all i s.t. $1 \leq i \leq n$), a more general solution η can be found by narrowing. Strictly speaking, the relative generality of substitution η w.r.t. ρ holds *modulo* $\mathcal{R}$ and is restricted to the variables of G, or more formally:

$$\eta \leq_{\mathcal{R}} \rho \; [\mathit{Var}(G)] \; (\textit{unification-completeness})$$

This means that there exists a substitution σ s.t., for all $x \in \mathit{Var}(G)$, the equation $x\rho = x\eta\sigma$ holds in $\mathcal{R}$, which can be proved by rewriting terms $x\rho$ and $x\eta\sigma$ in $\mathcal{R}$ to the same normal form, due to canonicity. The subindex $\mathcal{R}$ in $\leq_{\mathcal{R}}$ can be dropped only when we restrict our interest to *normalized* (or irreducible) substitutions, which is generally understood as a *weaker* result from both the semantic as well as the pragmatic point of view [*Meseguer and Thati*, 2007]. If we drop the termination of $\mathcal{R}$ while keeping confluence, narrowing is (unification-) complete only w.r.t. normalizable solutions [*Middeldorp and Hamoen*, 1994].

In the extensive literature about narrowing, unification-completeness has been thoroughly investigated for a number of narrowing restrictions which are obtained by imposing specific narrowing *strategies*; see [*Hanus*, 1994] for a survey. In this thesis, we restrict our interest to *ordinary* (sometimes called full, unrestricted, or simple) narrowing, as defined in Chapter 2. An investigation of completeness or termination for sophisticated narrowing strategies is beyond the scope of this thesis.

From a practical point of view, equational unification problems can be seen as a special case of reachability problems. Namely, under canonicity of $\mathcal{R}$, solving a unification problem $\exists \bar{x}.\ s = t$ can be transformed into solving the corresponding reachability problem $\exists \bar{x}.\ (s \approx t) \rightarrow^* \mathtt{true}$ in the extended term rewriting system $\mathcal{R} \cup (x \approx x \rightarrow \mathtt{true})$ where both problems have the same solutions provided that $\approx$ is a fresh binary function symbol and $\mathtt{true}$ is a fresh constant [*Meseguer and Thati*, 2007; *Middeldorp and Hamoen*, 1994]. The extension of $\mathcal{R}$ with the extra rule $(x \approx x \rightarrow \mathtt{true})$ allows treating equality $=$ as an ordinary function symbol $\approx$ and syntactic unification as a narrowing step, i.e., in the extended TRS, the "term" $s \approx t$ narrows to $\mathtt{true}$ with substitution σ iff σ is the most general unifier of s and t. Alternative formulations of narrowing-based equational unification procedures that do not extend $\mathcal{R}$ by this extra rewrite rule complement the narrowing calculus with an additional inference rule to cope with syntactic unification, e.g. [*Hölldobler*, 1989].

As stated above, the completeness of narrowing as a procedure to solve equational goals heavily depends on the condition that the rewrite rules are confluent. Actually, in the standard equational setting, confluence is the property which allows considering equations as rewrite rules (oriented from left to right). The equational theory axiomatized by $\{\mathtt{f(a)} = \mathtt{b}, \mathtt{f(a)} = \mathtt{c}\}$ is a trivial counter-example to unification-completeness when confluence does not hold. Here narrowing fails to prove the equation $\mathtt{b} = \mathtt{c}$ in the corresponding (oriented) TRS $\mathcal{R} = \{\mathtt{f(a)} \rightarrow \mathtt{b},\ \mathtt{f(a)} \rightarrow \mathtt{c}\}$, whereas $\mathtt{b} = \mathtt{c}$ holds in the original equational theory.

In [*Meseguer and Thati*, 2007], reachability goals $s_1 \rightarrow^* t_1, \ldots, s_n \rightarrow^* t_n$ are investigated in non-confluent term rewriting systems in order to solve verification problems of cryptographic protocols. Many safety properties (i.e., properties of a system that are defined in terms of certain events not happening) can be characterized in terms of reachability problems. By finding all solutions to a reachability goal $s \rightarrow^* t$ (i.e., the substitutions σ such that $\mathcal{R} \vdash s\sigma \rightarrow^* t\sigma$), the subset of the states denoted by s that can reach a subset of the states denoted by t can be easily inferred. Hence, reachability problems extend narrowing capabilities to a wider spectrum that includes the analysis of concurrent systems. Similarly to the equational case, the procedure for solving reachability goals performs syntactic unification at the last step of the derivation; this way, trivial goals such as $x \rightarrow^* y$ (where there is no redex to narrow) do succeed in computing a more general solution. In the reachability context, confluence is no longer a reasonable (or needed) assumption and is thus done away with (e.g., concurrent systems are inherently non-deterministic).

The new completeness results for narrowing given in [*Meseguer and Thati*, 2007][3] for solving reachability goals in (possibly) non-confluent TRS's are summarized as follows. Narrowing is *weakly* complete, i.e., complete w.r.t. *normalized* solutions: for every normalized solution ρ to a reachability goal G, a (syntactically) more general

[3]The completeness results in [*Meseguer and Thati*, 2007] concern more general rewrite theories that consist of a set of rewrite rules $\mathcal{R}$ together with a set of equations E so that rewriting and narrowing in $\mathcal{R}$ are defined *modulo* E.

solution η is found by narrowing, in symbols:

$$\eta \leq \rho\ [Var(G)]\ (\textit{weak reachability-completeness})$$

Note that neither confluence nor termination of $\mathcal{R}$ are required.

In [*Meseguer and Thati*, 2007], *strong* reachability-completeness (i.e., completeness w.r.t. not necessarily normalized solutions, i.e. solutions that can be further rewritten by $\mathcal{R}$) is proved to hold in the following two particular classes of TRS's: (i) topmost, and (ii) right-linear (provided that we additionally restrict ourselves to linear reachability goals $\wedge_{i=1}^{n}(s_i \to^* t_i)$, where each s_i is linear). Under these assumptions, for every solution ρ to a reachability goal G, a more general solution η (*modulo* $\mathcal{R}$) is computed by narrowing, i.e., $\eta \leq_{\mathcal{R}} \rho\ [Var(G)]$. In the reachability setting, where confluence cannot be assumed and thus equality in $\mathcal{R}$ cannot be decided by rewriting, the definition is translated as follows: there is a (syntactic) instance θ of the computed substitution η such that the (possibly not normalized) solution ρ reduces to θ. To be precise:

$$\rho\restriction_{Var(G)} \to^*_{\mathcal{R}} \theta\restriction_{Var(G)} \text{ and } \eta \leq \theta\ [Var(G)]\ (\textit{strong reachability-completeness})$$

Of course, unification-completeness trivially implies reachability-completeness, hence (strong) reachability-completeness of narrowing holds for canonical programs, whereas narrowing is not unification-complete in either right-linear or topmost TRS's [*Middeldorp and Hamoen*, 1994].

In the case of right-linear TRS's, linearity of the goal is a key requirement which cannot be dropped, as shown in the following example.

Example 1.2. [*Meseguer and Thati*, 2007] *Consider the TRS* $\mathcal{R} = \{\mathtt{f(b,c)} \to \mathtt{d}, \mathtt{a} \to \mathtt{b}, \mathtt{a} \to \mathtt{c}\}$. *The non-linear reachability goal* $\mathtt{f(x,x)} \to^* \mathtt{d}$ *has a solution* $\{x \mapsto \mathtt{a}\}$, *whereas there is no narrowing derivation stemming from the term* $\mathtt{f(x,x)}$.

This example shows that reachability-incompleteness of narrowing for general TRS's is mainly due to rewrites that must happen within non-normalized substitutions but are missed by the narrowing procedure, since narrowing steps do not apply to variable positions. In the standard equational setting, these "under the feet" rewritings are inconsequential, due to confluence.

1.1.2 Existing termination results for narrowing

In the literature, the termination of narrowing has received less attention than completeness. Actually, termination of narrowing is a much more difficult property to achieve than termination of standard term rewriting; see [*Ohlebusch*, 2002] for a survey on rewriting termination.

Termination results for narrowing calculi have been obtained as a by-product of other works that address the decidability of equational unification; a summary can be found in [*Dershowitz and Mitra*, 1999]. Most of these results are highly restrictive and do not allow any recursively defined function. Many introduce specially tailored equational unification procedures based on the generally more expensive "top-down decomposition approach" outlined in [*Martelli et al.*, 1986] (not considered in this thesis). Narrowing-based procedures with a finite search space often incorporate a test to cut unproductive, infinitely failing derivations [*Alpuente et al.*, 1995b; *Chabin and Réty*, 1991; *Dershowitz and Sivakumar*, 1988] or a kind of graph-based *memoization* technique [*Antoy and Ariola*, 1997; *Escobar and Meseguer*, 2007] to achieve, in some cases, a finite representation of an infinite narrowing space. There are popular[4] (syntactic) conditions that, together with termination and (often) confluence of $\mathcal{R}$, are required for the termination of these procedures. These include [*Dershowitz and Mitra*, 1999]: *left-linearity* (no variable appears in the lhs of a rewrite rule more than once); right-hand side (rhs) groundness, *right-groundness* (rhs's of rewrite rules contain no variable); and *left-flatness* (each argument occurring at the lhs of a rewrite rule is either a variable —often called *shallow* [*Comon et al.*, 1994]— or a ground term).

Unfortunately, the decidability of unification for a given equational theory does not imply the termination of ordinary narrowing in the corresponding TRS. For instance, unification is decidable in the equational theory associated to the function add of Example 1.1 (see e.g. [*Dershowitz and Mitra*, 1999]) whereas narrowing does not terminate for the input equation $\mathtt{add}(w,\mathtt{s}(\mathtt{0})) = \mathtt{0}$ (as we have shown). Achieving termination without losing completeness is possible for this particular example by adding an extra "failure rule", which is able to detect a clash conflict between the irreducible symbols 0 and s in the derived equational goal $\mathtt{s}(\mathtt{add}(x,\mathtt{s}(\mathtt{0}))) = \mathtt{0}$. However, as the following example shows it is more difficult in general.

Example 1.3. *Consider the TRS consisting of the "shallow" oriented commutativity axiom for a binary symbol f:* $\mathcal{R} = \{\mathtt{f}(\mathtt{x},\mathtt{y}) \to \mathtt{f}(\mathtt{y},\mathtt{x})\}$*. An extra artifact such as a "loop checker" would be needed to stop narrowing from the input equation* $\mathtt{f}(\mathtt{x},\mathtt{y}) = \mathtt{z}$ *in* $\mathcal{R}$*, whereas the corresponding equational theory defined by* $\mathcal{R}$ *is not only decidable but actually finitary [Siekmann, 1989b] (actually, the considered equational goal has exactly the one solution (modulo* $\mathcal{R}$*)* $\{z \mapsto \mathtt{f}(\mathtt{x},\mathtt{y})\}$*.*

Summarizing, the only positive result in the literature concerning the termination of ordinary narrowing was proved in [*Christian*, 1992] and holds for every left-flat TRS $\mathcal{R}$ that is compatible with a termination ordering $<$. Termination of narrowing does not hold for systems with flat right-hand sides (even if linearity is also imposed), as proved in [*Mitra and Dershowitz*, 1996].

In general, whenever the lhs of a rewrite rule is not flat, aliasing due to repeated variables can cause troublesome propagation of hazardous structure as shown by the following example.

Example 1.4. *[Christian, 1992] The non-flat rule* $\mathtt{f}(\mathtt{f}(\mathtt{x})) \to \mathtt{x}$ *is "safe" when used to narrow a linear term like* $\mathtt{c}(\mathtt{f}(\mathtt{u}),\mathtt{v})$*: it produces the term* $\mathtt{c}(\mathtt{x},\mathtt{v})$*, which cannot be further narrowed. However, the non-linear term* $\mathtt{c}(\mathtt{f}(\mathtt{x}),\mathtt{x})$ *can be narrowed indefinitely:*

$$\mathtt{c}(\underline{\mathtt{f}(\mathtt{x})},\mathtt{x}) \leadsto_{\{x\mapsto \mathtt{f}(\mathtt{x}')\}} \mathtt{c}(\mathtt{x}',\underline{\mathtt{f}(\mathtt{x}')}) \leadsto_{\{x'\mapsto \mathtt{f}(\mathtt{x}'')\}} \mathtt{c}(\underline{\mathtt{f}(\mathtt{x}'')},\mathtt{x}'')\cdots$$

[4]These properties have been studied in the context of other rewriting-related properties and problems also, such as joinability, modularity of termination, and modularity of confluence.

1.2 Plan of the thesis

In this thesis we make several contributions related to the termination of narrowing. In the first part we focus on the study of the termination of narrowing and *basic* narrowing. In Chapter 3, we focus on the syntactic restrictions (*classes*) of TRSs where narrowing terminates, learning numerous insights on the way. Next, in Chapter 4 we study the shape of infinite narrowing derivations, which leads to a method based on the technique of Dependency Pairs [*Arts and Giesl*, 2000] for automated proofs of termination of narrowing. In Chapter 5 we consider automated proofs of termination of narrowing *from an initial query*. These two chapters consider two independent extensions of the Dependency Pair technique which can be complementary.

In the second part of the thesis, we study the modularity properties of the termination of *basic* narrowing, paving the way for the modular combination of theories for equational unification with basic narrowing. Relying on these results, we develop the modular combination of theories for equational unification with basic narrowing.

To be precise, the thesis contains the following contributions:

(i) For many applications it can be of interest to know whether narrowing terminates for a given *class* of TRSs, instead of for a particular TRS. These classes are often specified as syntactic restrictions. We ascertain several classes of TRSs where narrowing *always* terminates, providing syntactic criteria there where possible.

(ii) *Basic* narrowing [*Hullot*, 1980] is a restriction of narrowing with numerous applications in the area of equational unification. We study existing results on the termination of *basic* narrowing and generalize the termination criterion of [*Hullot*, 1980] by dropping the superfluous requirement of canonicity (i.e. termination + confluence). In addition, we show how to relate termination of narrowing with termination of basic narrowing, and give a criterion for the termination of narrowing in terms of the termination of basic narrowing.

(iii) By studying the shape of infinite narrowing derivations, we generalize the dependency pair approach of [*Arts and Giesl*, 2000] to produce automated proofs of termination of narrowing. For this we need to consider a new type of dependency pairs, *ll*-dependency pairs, as well as a notion of chain specific to narrowing. Moreover, we show how to translate termination of narrowing into termination of rewriting, so the wealth of techniques available to prove the termination of rewriting can be reused to prove the termination of narrowing. Finally, our technique is generally useful in that it can be applied to *any* class of TRSs, not being restricted to particular classes of systems.

(iv) In many applications, termination for *all* possible queries is not required. Instead, it suffices that narrowing terminates for a given *class* of queries. Indeed, even if the TRS is not terminating in general, the set of queries considered can be terminating. This problem has appeared in previous work in the literature, e.g.

for the termination of logic programs [*Schneider-Kamp et al.*, 2009b], the termination of Haskell programs [*Giesl et al.*, 2006a], or the termination of narrowing [*Vidal*, 2008].

We extend the dependency pair approach to consider termination problems from an initial goal. We do so in a way which is independent of the underlying relation, be it rewriting, narrowing or any other relation as e.g. innermost rewriting. Since our method is not tied to narrowing in any way it can be reused for other purposes.

(v) Given two TRSs $\mathcal{R}$ and $\mathcal{P}$, we speak of *relative* termination of $\mathcal{R}$ with regard to $\mathcal{P}$ when there are no infinite $\to^*_{\mathcal{P}} \circ \to_{\mathcal{R}} \circ \to^*_{\mathcal{P}}$ derivations with an infinite number of $\to_{\mathcal{R}}$ steps. We show how to use the dependency pair approach to produce proofs of relative termination when $\mathcal{R}$ and $\mathcal{P}$ form a hierarchical combination (i.e. no rules of $\mathcal{P}$ contain calls to functions defined in $\mathcal{R}$). This means that the wealth of techniques available for rewriting termination can be applied directly to prove relative termination when the mentioned condition holds.

(vi) We put together the frameworks for initial goal termination and relative termination in order to prove termination of narrowing from an initial goal, by recasting it as a problem of relative termination w.r.t. a set of *generators*, following [*Vidal*, 2008]. Our work is a significant extension of [*Vidal*, 2008], due to the improved generality —we manage to lift the usual *left-linearity* restriction—, power —as demonstrated by our empirical benchmarks—, and efficiency —since our method replaces heuristics and unstructured search by constraint solving— of the resulting technique.

(vii) As an immediate application of the above, narrowing can be used to prove the termination of functional-logic and logic programs. We briefly develop the latter on top of the transformational approach of [*Schneider-Kamp et al.*, 2009b]. The resulting method for termination of logic programs is potentially more powerful than the original, as our benchmarks make explicit.

(viii) To analyze the termination properties of the combination of TRSs, we study the modularity properties of termination of basic narrowing, unveiling its highly modular nature.

(ix) Pairing the modularity results for the termination of basic narrowing uncovered above with existing results in the literature about the modularity of its completeness, we distill a number of results about the modularity of the decidability of equational unification, allowing for the modular combination of theories that satisfy the conditions we unveil.

Major parts of these contributions have already been published as joint work in international journals ([*Alpuente et al.*, 2010a, b, 2009]) and in conference proceedings ([*Alpuente et al.*, 2008a, b; *Iborra et al.*, 2010]). However, this thesis contains a number of contributions and improvements over previous developments which are entirely original. These are enumerated individually at the beginning of every chapter.

Many of the techniques presented in this thesis can be efficiently automated. The two techniques for automated proofs of termination of narrowing, corresponding to Chapters 4 and 5, have been implemented in the tool [Narradar], consisting of over 10.000 lines of Haskell code developed solely by the author. NARRADAR itself makes use of other tools, including modern SMT solvers as [Yices] and, in earlier versions, of the termination tool AProVE [*Giesl et al.*, 2004].

The thesis is organized as follows. First, we recall the foundations of term rewriting and narrowing in Chapter 2. Then, Chapter 3 introduces Hullot's *basic* restriction of narrowing, with applications in equational unification due to its good completeness and termination properties. The latter are reviewed and improved, by dropping a superfluous canonicity requirement. Then a faulty result connecting the termination of basic narrowing with the termination of narrowing is reviewed and and a corrected result is provided (contribution (i)). From here, several stages of generalizations yield a number of syntactic classes (contribution (ii)) where narrowing is complete and terminating. We then move to the domain of automatic methods which are not limited to syntactic classes. In chapter 4, we study the shape of minimally non-terminating narrowing derivations, and extract a notion of narrowing dependency pairs that effectively models them. On top of this notion we build our termination criterion, and turn it into an automatic method in the *narrowing* dependency pair framework (contribution (iii)). Then in Chapter 5 we consider the problem of termination *from an initial goal*, and develop several extensions to the dependency pair approach, in order to deal with initial goal problems and relative termination problems (contributions (iv) and (v)). These extensions are put to good use in the second part of this Chapter for proving the termination of narrowing from an initial goal (contribution (vi)). After that, we briefly discuss how these results can be applied to proving the termination of logic programs (contribution (vii)).

Chapter 7 is devoted to the study of the modularity properties of the termination of basic narrowing (viii) in several classes of modular decompositions, including hierarchical combinations. We show that termination of basic narrowing is modular up to *proper* hierarchical combinations, a very general decomposition class. These results are then complemented in Chapter 8 with a survey of the results on modularity of completeness of basic narrowing in the literature. It is possible now to extract modularity results for completeness *plus* termination, which together ensure the decidability of equational unification (ix). The thesis concludes with a discussion of the lines of future work in Chapter 9.

2
Foundations

In this section we recall the standard notions and terminology of term rewriting. Readers familiar with these can skip this chapter, while other readers might find it more useful to consult a reference book such as [*Baader and Nipkow*, 1998] or [*Ohlebusch*, 2002].

Relations

Given a binary relation $\Rightarrow$, we denote by $\Rightarrow^+$ the transitive closure of $\Rightarrow$ and by $\Rightarrow^*$ its reflexive and transitive closure. Thus $t \Rightarrow^* s$ means that t can be reduced to s in zero or more steps. Given two relations $\Rightarrow_1$ and $\Rightarrow_2$, their composition $\Rightarrow_1 \circ \Rightarrow_2$ is defined by $\{(x, z) \mid \exists y.(x, y) \in \Rightarrow_1 \wedge (y, z) \in \Rightarrow_2\}$. A $\Rightarrow$ *derivation* is a (possibly empty) sequence of $\Rightarrow$ steps. A relation is well-founded if there are no infinite sequences.

Terms, variables and positions

A *signature* Σ is a set of function symbols each of which has a fixed arity. We often write $\mathsf{f}/n \in \mathcal{F}$ to denote that the arity of function f is n. Given a countably infinite set of variables $\mathcal{V}$ with $\Sigma \cap \mathcal{V} = \emptyset$, we denote the domain of *terms* by $\mathcal{T}(\Sigma, \mathcal{V})$. We assume that Σ always contains at least one constant $\mathsf{f}/0$. We use $\mathsf{f}, \mathsf{g}, \ldots$ to denote functions and $x, y, \ldots$ to denote variables. Terms are viewed as labelled trees in the usual way, where $\mathcal{T}(\Sigma, \mathcal{V})$ and $\mathcal{T}(\Sigma)$ denote the non-ground and the ground term algebra built on $\Sigma \cup \mathcal{V}$ and Σ, respectively. Syntactic equality between terms is denoted by $\equiv$.

Positions are defined as sequences of natural numbers used to address subterms of a term, with ϵ as the root (or top) position (i.e., the empty sequence). Concatenation of positions p and q is denoted by $p.q$, and $p < q$ is the usual prefix ordering, with $\epsilon \leq p$ for all positions p. Two positions p, q are disjoint, denoted by $p \parallel q$, if neither $p < q$, $p > q$, nor $p = q$.

The root symbol of a term is denoted by $root(t)$. Given $S \subseteq \Sigma \cup \mathcal{V}$, $\mathcal{P}os_S(t)$ denotes the set of positions of a term t that are rooted by function symbols or variables in S. $\mathcal{P}os_{\{f\}}(t)$ with $f \in \Sigma \cup \mathcal{V}$ will be simply denoted by $\mathcal{P}os_f(t)$, and $\mathcal{P}os_{\Sigma\cup\mathcal{V}}(t)$ will be simply denoted by $\mathcal{P}os(t)$. $t|_p$ is the subterm at the position p of t. $t[s]_p$ is the term t with the subterm at the position p replaced with term s. We say that u is a *proper* subterm of t if $u = t|_p$ and $p < \epsilon$. We write $t \trianglerighteq u$ or $u \trianglelefteq t$ if u is a subterm of t, and $t \rhd u$ or $u \lhd t$ if u is a *proper* subterm of t.

By $\mathit{Var}(s)$, we denote the set of variables occurring in the syntactic object s. A *fresh* variable is a variable that appears nowhere else. A *linear* term is one where every variable occurs only once. A *ground* term is one where no variable occurs.

A context is a term with several occurrences of a special fresh symbol $\Box$. If $C[\,]$ contains k occurrences of $\Box$ at positions $p_1, \ldots, p_k$, we write $C[t_1]_{p_1}[t_2]_{p_2} \ldots [t_k]_{p_k}$, or simply $C[t_1, \ldots, t_k]$, to denote the term $(C[t_1]_{p_1}) \cdots [t_k]_{p_k}$.

Substitutions and unifiers.

A *substitution* $\sigma : \mathcal{V} \to \mathcal{T}(\Sigma, \mathcal{V})$ is a mapping from the set of variables $\mathcal{V}$ into the set of terms $\mathcal{T}(\Sigma, \mathcal{V})$ with a (possibly infinite) domain which can be homomorphically extended to a morphism from terms to terms. A substitution is represented as $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$ for variables $x_1, \ldots, x_n$ and terms $t_1, \ldots, t_n$. The *domain* of a substitution σ is $D(\sigma) = \{x \in \mathcal{V} \mid x\sigma \neq x\}$, and $\mathit{Rng}(\sigma) = \{x\sigma \mid x \in D(\sigma)\}$ is its *range*. The set of variables in $\mathit{Rng}(\sigma)$ is denoted by $\mathit{VRng}(\sigma)$. A finite substitution σ has a finite domain $D(\sigma)$. The *empty substitution* is denoted by id, i.e., $D(\mathit{id}) = \emptyset$. The application of a substitution θ to a term t is denoted by $t\theta$, using postfix notation. Composition of substitutions is denoted by juxtaposition, i.e., the substitution $\sigma\theta$ denotes $(\theta \circ \sigma)$, i.e., $\sigma\theta(x) = \theta(\sigma(x))$.

We write $\theta\!\upharpoonright_{\mathit{Var}(s)}$ to denote the restriction of the substitution θ to the set of variables in s; by abuse of notation, we often simply write $\theta\!\upharpoonright_s$. A substitution θ is more general than σ, denoted by $\theta \leq \sigma$, if there is a substitution γ such that $\theta\gamma = \sigma$. Given a set of variables W, we write $\theta = \nu\ [W]$ (resp. $\theta \leq \nu\ [W]$) for $\theta\!\upharpoonright_W = \nu\!\upharpoonright_W$ (resp. $\theta\!\upharpoonright_W \leq \nu\!\upharpoonright_W$).

A *unifier* of terms s and t is a substitution ϑ such that $s\vartheta = t\vartheta$. The *most general unifier* of terms s and t, denoted by $\mathit{mgu}(s, t)$, is a unifier θ such that for any other unifier θ', $\theta \leq \theta'$.

A substitution σ is idempotent if for every term t, $\sigma(\sigma(t)) = \sigma(t)$. We say that two idempotent substitutions θ_1 and θ_2 are *compatible* if their correponding bindings "unify", that is, there is θ s.t. $x\theta_1\theta \equiv x\theta_2\theta$, for all $x \in D(\theta_1) \cap D(\theta_2)$.

Term Rewriting Systems

A *term rewriting system* (TRS) $\mathcal{R}$ is a pair (Σ, R), where R is a finite set of rewrite rules of the form $l \to r$ such that $l, r \in \mathcal{T}(\Sigma, \mathcal{V})$, $l \notin \mathcal{V}$, and $\mathit{Var}(r) \subseteq \mathit{Var}(l)$ (the latter is often referred to as the variable condition). We will often write just $\mathcal{R}$ or (Σ, R) instead of $\mathcal{R} = (\Sigma, R)$. If we omit the variable condition, i.e . $\mathit{Var}(r) \subseteq \mathit{Var}(l)$ for every rule $l \to r$, then we talk of *generalized* TRSs (GTRS) instead. Variables in the right hand side which do not occur in the left hand side are called *extra* variables. We denote the extra variables of a rule $l \to r$ by $\mathcal{E}\mathit{Var}(l \to r)$, or simply $\mathcal{E}\mathit{Var}(r)$ when the rule is known from the context.

Given a (generalized) TRS $\mathcal{R} = (\Sigma, R)$, the signature Σ is often partitioned into two disjoint sets $\Sigma = \mathcal{C} \uplus \mathcal{D}$, where $\mathcal{D} = \{f \mid f(t_1, \ldots, t_n) \to r \in R\}$ and $\mathcal{C} = \Sigma \setminus \mathcal{D}$.

Note that $\mathcal{D} \cap \mathcal{C} = \emptyset$ by definition. Symbols in $\mathcal{C}$ are called *constructors*, and symbols in $\mathcal{D}$ are called *defined functions*. The elements of $\mathcal{T}(\mathcal{C}, \mathcal{V})$ are called *constructor terms*. We often represent a (generalized) TRS as $\mathcal{R}(\mathcal{D}, \mathcal{C}, R)$. We also use the notation $\Sigma = \mathcal{D} \uplus \mathcal{C}$ to point out that $\mathcal{D}$ are the defined function symbols and $\mathcal{C}$ are the constructors of a signature Σ. A substitution σ is (ground) *constructor*, if $x\sigma$ is a (ground) constructor term for all $x \in D(\sigma)$.

The rewriting relation $\to_{\mathcal{R}}$ is a binary relation on terms defined by $\to_{\mathcal{R}}= (C[l\sigma]_p, C[r\sigma]_p)$ where C is a context, p a position, σ a substitution and $\mathcal{R}$ a (generalized) TRS with $l \to r \in \mathcal{R}$. We often write $\to^{p}_{l \to r}$ to make the position and rule used explicit. We often write $\overset{>\epsilon}{\to}_{\mathcal{R}}$ to denote steps in which the selected redex happens below the root.

When no confusion can arise, we omit the subscript $\mathcal{R}$ in $\to_{\mathcal{R}}$. We also omit the reduced position p when it is not relevant. A term is a reducible expression or *redex* if it is an instance of the left hand side of a rule in $\mathcal{R}$. A term s is a *normal form* with regard to a relation $\Rightarrow$ (or simply a normal form where no confusion can arise), if there is no term t such that $s \Rightarrow t$. A term s is a *head normal form* with regard to a TRS $\mathcal{R}$ if there are no terms t, t' s.t. $s \to^*_{\mathcal{R}} t' \overset{\epsilon}{\to}_{\mathcal{R}} t$. A substitution σ is *normalized* with regard to a relation $\Rightarrow$ if, for every $x \in \mathcal{V}$, $x\sigma$ is a normal form with regard to $\Rightarrow$. A substitution is said to be *normalizable* with regard to a relation $\Rightarrow$ if, for every $x \in \mathcal{V}$, $x\sigma$ has a normal form with regard to $\Rightarrow$.

Popular TRS classifications

A (generalized) TRS $\mathcal{R}$ is called *left–linear* (respectively *right–linear*) if, for every $l \to r \in \mathcal{R}$, l (respectively r) is a linear term. A *linear* (generalized) TRS is both left and right–linear.

A constructor system is a (generalized) TRS whose left-hand sides are *patterns*, i.e., terms of the form $f(c_1, \ldots, c_k)$ where $f \in \mathcal{D}$ and $c_1, \ldots, c_k$ are constructor terms. A term whose root symbol is a defined function is called *root-defined*.

A (generalized) TRS $\mathcal{R}$ is called *conservative* (or regular) if, for every $l \to r \in \mathcal{R}$, $Var(l) = Var(r)$.

A GTRS $\mathcal{R}$ is *confluent* if, whenever $t \to^*_{\mathcal{R}} s_1$ and $t \to^*_{\mathcal{R}} s_2$, there exists a term w s.t. $s_1 \to^*_{\mathcal{R}} w$ and $s_2 \to^*_{\mathcal{R}} w$. A confluent and terminating TRS is called canonical[1]. In canonical TRS's, each term has one (and only one) normal form.

A TRS $\mathcal{R}$ is *weakly normalizing* if every term has a normal form that can be reached by a finite rewriting sequence.

Two (possibly renamed) rules $l \to r$ and $l' \to r'$ *overlap* if there is $p \in \mathcal{P}os_{\Sigma}(l)$ and substitution σ such that $l|_p\sigma \equiv l'\sigma$. The pair $\langle l\sigma[r'\sigma]_p, r\sigma \rangle$ is called a *critical pair*; it is called an *overlay* if $p \equiv \epsilon$. A critical pair $\langle t, s \rangle$ is *trivial* if $t \equiv s$. A left-linear TRS without critical pairs is called *orthogonal*. A left-linear TRS whose critical pairs are trivial overlays is called *almost orthogonal* or *overlay system*. Note that orthogonal

[1]Canonical TRS's are sometimes called *complete* [*Knuth and Bendix*, 1970; *Hullot*, 1980; *Middeldorp and Hamoen*, 1994].

TRS's are almost orthogonal and almost orthogonality implies confluence [TeReSe, 2003].

A TRS $\mathcal{R}$ is called *topmost* if, for every term of interest t, all rewritings on t are performed at the root position of t. Topmost TRS's are a central part of the dependency pair approach to the termination of narrowing, where to decide the termination of a TRS $\mathcal{R}$ one studies instead the termination of a topmost TRS derived from $\mathcal{R}$. For example, given the encoding of addition in Example 1.1, one extracts a derived TRS given by the rule

$$\text{ADD}(\mathtt{0}, y) \to \text{ADD}(x, y) \tag{2.1}$$

In this case the terms of interest are those which do not contain calls to ADD below the root position. Topmost TRS's are also relevant in programming languages. For instance, in Haskell [Peyton Jones, 2003] or Maude [Clavel et al., 2007], rewrite rules can be defined so that the type (or sort) information forces rewrites to happen only at the top of terms. In this case, the terms of interest are those which are *well sorted*.

In Maude, it is also possible to introduce freezing specifications that block rewrites at any proper subterm position. Actually, many concurrent systems of interest, including the vast majority of distributed algorithms, admit quite natural topmost specifications [Meseguer and Thati, 2007].

Narrowing

Let $\mathcal{R}$ be a (generalized) TRS. Let $l \to r \ll \mathcal{R}$ denote that $l \to r$ is a new variant of a rule in $\mathcal{R}$ such that $l \to r$ contains only *fresh* variables, i.e., contains only variables not previously met.

The narrowing relation $\rightsquigarrow_{\mathcal{R}}$ is a binary relation on terms defined by $\rightsquigarrow_{\mathcal{R}} = (C[t]_p, C\sigma[r\sigma]_p)$ where C is a context, p a position, $\mathcal{R}$ a (generalized) TRS with $l \to r \ll \mathcal{R}$ and σ a most general unifier of t and l. We often write $\rightsquigarrow^{p}_{\sigma,l\to r}$ to make explicit the position, rule and substitution computed.

A *narrowing derivation* $t_0 \rightsquigarrow^{*}_{\sigma} t_n$ denotes a sequence of narrowing steps $t_0 \rightsquigarrow_{\sigma_1} \ldots \rightsquigarrow_{\sigma_n} t_n$ with $\sigma = \sigma_n \circ \cdots \circ \sigma_1$ (if $n = 0$ then $\sigma = id$). We write $s \rightsquigarrow^{*}_{\sigma,\mathcal{R}} t$ if there exists a narrowing derivation $s \rightsquigarrow_{\sigma_1,\mathcal{R}} \circ \rightsquigarrow_{\sigma_2,\mathcal{R}} \cdots \rightsquigarrow_{\sigma_n,\mathcal{R}} t$ such that $\sigma = \sigma_1\sigma_2 \ldots \sigma_n$. We say a term t is a narrex for $\mathcal{R}$ if there is a term t' s.t. $t \rightsquigarrow_{\mathcal{R}} t'$.

The following important result, originally enunciated by Hullot, establishes the correspondence between narrowing and rewriting derivations. It is also known as the soundness theorem or the lifting lemma. Concretely, it states that every rewriting derivation can be projected as a narrowing derivation, while every narrowing derivation projects a certain set of rewriting derivations.

Proposition 2.1 (Lifting Lemma [Hullot, 1980])**.** *, Let $\mathcal{R}$ be a TRS, t_0 be a term, and consider a narrowing derivation starting from t_0:*

$$t_0 \overset{p_0}{\rightsquigarrow}_{\sigma_0,l_0\to r_0} t_1 \overset{p_1}{\rightsquigarrow}_{\sigma_1,l_1\to r_1} \cdots \overset{p_n}{\rightsquigarrow}_{\sigma_n,l_n\to r_n} t_n \tag{2.2}$$

Define a substitution $\theta = \sigma_0\sigma_1 \ldots \sigma_n$. There exists an associated rewriting sequence starting from $t_0\theta$:

$$t_0\theta \xrightarrow{p_0}_{l_0\to r_0} t_1\theta \xrightarrow{p_1}_{l_1\to r_1} \ldots \xrightarrow{p_n}_{l_n\to r_n} t_n\theta \tag{2.3}$$

Conversely, to every rewriting derivation 2.3 and every normalized substitution θ, we can associate a narrowing derivation 2.2. More precisely, if θ is a normalized substitution and $s\theta \to^*_{\mathcal{R}} t'$, then there exist a term s' and substitutions θ' and σ such that:

- $s \rightsquigarrow^*_{\sigma,\mathcal{R}} s'$,
- $s'\theta' = t'$,
- $\sigma\theta' = \theta[s]$,
- *θ' is normalized.*

and one can assume that the narrowing derivation uses the same rules at the same positions as the rewriting derivation.

Below we extend the lifting lemma for GTRSs with extra variables. As in the case for TRSs, every narrowing derivation can be projected as a set of rewriting derivations, but the converse direction does not hold.

Proposition 2.2 (Lifting Lemma extended)**.** *, Let $\mathcal{R}$ be a GTRS, t_0 be a term, and consider a narrowing derivation starting from t_0:*

$$t_0 \overset{p_0}{\rightsquigarrow}_{\sigma_0,l_0\to r_0} t_1 \overset{p_1}{\rightsquigarrow}_{\sigma_1,l_1\to r_1} \ldots \overset{p_n}{\rightsquigarrow}_{\sigma_n,l_n\to r_n} t_n \tag{2.4}$$

Define a (possibly infinite) substitution $\theta = \sigma_0\sigma_1 \ldots \sigma_n$. There exists an associated rewriting sequence starting from $t_0\theta$:

$$t_0\theta \xrightarrow{p_0}_{l_0\to r_0} t_1\theta \xrightarrow{p_1}_{l_1\to r_1} \ldots \xrightarrow{p_n}_{l_n\to r_n} t_n\theta \tag{2.5}$$

Proof. The original proof in *[Hullot, 1980, Theorem 1]* (or the more precise proof given in [*Middeldorp and Hamoen*, 1994]) applies with no changes to the case of extra variables. □

Termination

A TRS $\mathcal{R}$ is $(\to_{\mathcal{R}})$-terminating[2] if the relation $\to_{\mathcal{R}}$ is well-founded. A GTRS $\mathcal{R}$ is $(\rightsquigarrow_{\mathcal{R}})$-terminating if the relation $\rightsquigarrow_{\mathcal{R}}$ is well-founded.

An object t is said to be $(\Rightarrow)$-terminating, or just terminating when no confusion can arise, if there is no infinite $\Rightarrow$ sequence starting from t. Thus a term $t \in \mathcal{T}(\Sigma, \mathcal{V})$ is $(\to_{\mathcal{R}})$-terminating (resp. $(\rightsquigarrow_{\mathcal{R}})$-terminating) if there is no infinite rewriting (resp. narrowing) derivation in $\mathcal{R}$ starting from t. A set of objects is said to be $(\Rightarrow)$-terminating if every member object is $(\Rightarrow)$-terminating.

[2]also called strongly normalizing or noetherian

Proposition 2.3 (Termination of narrowing implies termination of rewriting)**.** *Let $\mathcal{R}$ be a TRS. If $\rightsquigarrow_{\mathcal{R}}$ is terminating, then $\to_{\mathcal{R}}$ is terminating.*

Proof. Immediate from the soundness of narrowing, i.e. Proposition 2.1. □

3 Syntactic Termination

Due to non-termination, narrowing behaves as a semi-decision procedure for the problem of equational unification in a wide variety of equational theories, as mentioned before. For instance, recall the equational theory for Peano addition from the introduction in Chapter 1, which was defined by the rules:

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

narrowing allows us to prove that the formula $\exists w \exists z$ s.t. $\mathtt{add}(w, \mathtt{s}(\mathtt{0})) = \mathtt{s}(\mathtt{s}(z))$ holds by computing the solution $\{w \mapsto \mathtt{s}(\mathtt{0}),\ z \mapsto \mathtt{0}\}$, whereas it cannot prove that the formula $\exists w$ s.t. $\mathtt{add}(w, \mathtt{s}(\mathtt{0})) = \mathtt{0}$ does not hold.

Under appropriate conditions, narrowing is complete as an equational unification algorithm as well as a procedure to solve *reachability* problems, as explained in the introduction. For instance, narrowing computes the solution $\{w \mapsto \mathtt{s}(z)\}$ for the reachability problem $\exists w \exists z$ s.t. $\mathtt{add}(\mathtt{0}, w) \to^* \mathtt{s}(z)$.

In this chapter, we are interested in identifying classes of TRSs where narrowing terminates and is still complete for solving reachability and equational unification problems. We do not consider extra artifacts to reduce or limit the narrowing space, such as restrictions or strategies. As such, we talk of unrestricted, ordinary or *full* narrowing indistinctly. We proceed as follows:

1. We extend an existing termination result, originally formulated for Basic Narrowing by [*Hullot*, 1980], to Full Narrowing in canonical TRSs. This is achieved by requiring the TRS to satisfy Réty's maximal commutation conditions, which allow the establishment of a correspondence between ordinary and basic narrowing derivations (Theorem 3.13).

   We complete this result by explicitly dropping the superfluous requirement of canonicity from Hullot's result, as the cognoscenti already tacitly do, generalizing Theorem 3.13 in Corollary 3.18. However the class of TRSs satisfying the maximal commutation conditions is quite restrictive, as it requires that the TRS is linear and, roughly speaking, constructor based.

2. From Corollary 3.18, we extract a practical criterion for the termination of narrowing that does not require confluence of the TRS, nor a termination ordering. We achieve this by imposing syntactic conditions on the TRS, namely that it be linear and rnf-*based*, a novel class of TRSs that can be seen as a generalization of left-linear constructor systems and that satisfy Réty's normalization condition (Corollary 3.27).

   A TRS is rnf-*based* if each argument occurring in the lhs of every rewrite rule is "unnarrowable", called *rigid normal form* (rnf), i.e., it contains no subterm that unifies with the lhs of any rule. The class of rnf-based TRSs includes both constructor systems and almost orthogonal TRSs as a particular case.

3. Then, we consider the class of TRSs where narrowing is strongly complete as a procedure to solve reachability goals. This allows us to prove narrowing termination in a number of TRSs where right-linearity is not explicitly required (Corollary 3.37).

4. Inspired by Christian's termination result [*Christian*, 1992], we further improve our results getting rid of left-linearity, proving termination for the subclass of *left-plain* TRSs, a novel class where arguments of the lhs's can be either ground or rnf-patterns (Theorem 3.61).

5. Finally, by using the known results for the strong reachability completeness of narrowing recently given by [*Meseguer and Thati*, 2007], we identify several almost purely syntactical, non-trivial classes of TRSs where narrowing has a finite search space and is still (strongly) complete as a procedure to solve reachability goals (Corollary 3.62).

   From the above results, termination of several popular classes of TRSs follows, including right-rnf TRSs which are either (i) almost orthogonal, (ii) constructor and either right-linear or confluent, (iii) topmost, or (iv) right-linear and the initial term is linear. These results are particularly practical since many interesting TRSs fit into one of these classes. Differently from Christian's criterion [*Christian*, 1992], our termination criteria do not resort to termination orderings, and are thus simpler to check.

Structure of the chapter

Section 3.1 introduces the *basic* refinement of narrowing. We recall Hullot's termination criterion for basic narrowing in canonical TRSs in Section 3.1.1. and derive a novel termination criterion for full narrowing employing Réty's maximal commutation property [*Réty*, 1987]. Then, in Section 3.2, we show that canonicity is a superfluous requirement in Hullot's result. This leads to a practical termination criterion for full narrowing in linear, rnf-based TRSs. Section 3.3 introduces the class of reachability-complete TRSs, which allows us to get rid of right-linearity. Finally, Section 3.4 provides a strong narrowing termination criterion which holds in left-plain,

Restrictions on $\mathcal{R}$	Reference
LF + cT	**[*Christian*, 1992, Lemma 2]**
RL + (LL or Co) + NC + bnT	**Thm. 3.13 (Hullot's result gen.)**
RL + (LL or Co) + NC + R-bnT	**Cor. 3.18 (Hullot's result repaired)**
R-rnf + L + rnf-B	Corollary 3.27
	e.g. R-rnf + L + CS
RL + R-rnf (+linear term)	**Corollary 3.56**
LP + RC + R-rnf	**Corollary 3.61**
R-rnf + LP + C	Corollary 3.62
	e.g. R-rnf + (either aO or CS + C)
R-rnf + Tp	Corollary 3.62
RL + (LL or Co) + NC + St	**Thm. 3.13, by [*Nieuwenhuis*, 1996]**

Legend

C	confluent	LL	left-linear	RL	right-linear
Tp	topmost	Co	conservative	CS	constructor system
R-rnf	right-rnf	rnf-B	rnf-based	LP	left-plain
LF	left-flat	L	linear	aO	almost Orthogonal
RC	reachability-complete	bnT	basic narrowing terminates		
NC	Rety's normalization condition				
R-bnT	all basic narrowing derivations starting from rule rhs's terminate				
St	standard theories saturated by basic paramodulation				
cT	compatible with a termination ordering				

Figure 3.1: Criteria for Narrowing termination

right-rnf TRSs, provided they are also reachability-complete. Figure 3 summarizes all the results included in this chapter.

The contents in this chapter have been published in [*Alpuente et al.*, 2009].

3.1 Basic Narrowing

Basic narrowing [*Hullot*, 1980] is a refinement of narrowing that restricts narrowing steps to a set of unblocked (or basic) positions and is still complete for equational unification in canonical TRSs, i.e., terminating and confluent TRSs.

Termination of basic narrowing was first studied by Hullot in [*Hullot*, 1980], where an erroneous termination result for narrowing was enunciated. The result incorrectly states that ordinary narrowing terminates in canonical TRSs when all *basic* narrowing derivations issued from the right hand side of each rewrite rule terminate. This can be refuted by the following counterexample.

Example 3.1. *Consider again the TRS* $\{\mathtt{f}(\mathtt{f}(x)) \to x\}$ *of Example 1.4, which is canonical and trivially satisfies the requirement that (basic) narrowing terminates for the rhs* x*. However, Example 1.4 above shows that an infinite narrowing derivation exists in* $\mathcal{R}$*.*

Actually, under the conditions established by Hullot's proof, nothing beyond the termination of basic narrowing can be concluded, as implicitly[1] corrected in Hullot's thesis [1981]. Note that basic narrowing does "safely" handle the TRS of Example 1.4 and blocks the infinite narrowing derivation after the first step.

We formulate basic narrowing using the original definition, given by Hullot and subsequently used by [*Réty*, 1987; *Middeldorp and Hamoen*, 1994], which is based on restricting narrowing steps to a distinguished set of *basic* positions.

Definition 3.2 (Basic Narrowing Derivation). *Given a narrowing derivation D:* $t_0 \overset{p_1}{\leadsto}_{\theta_1,\mathcal{R}} t_1 \overset{p_2}{\leadsto}_{\theta_2,\mathcal{R}} \cdots \overset{p_n}{\leadsto}_{\theta_n,\mathcal{R}} t_n$*, where* $l_i \to r_i \in R$ *is used at step* i*, we inductively define the* basic positions *of* D *as:*

$$B_0 = \mathcal{P}os_\Sigma(t_0)$$
$$B_i = (B_{i-1} \setminus \{p_i.p \mid p \in \mathcal{P}os(t_{i-1}|_{p_i})\}) \cup p_i.\mathcal{P}os_\Sigma(r_i)$$

Informally, a basic occurrence is a non-variable occurrence of the original term or one that was introduced by the non-variable content of the rhs of an applied rule.

We define a basic narrowing derivation $s \leadsto^*_\theta t$ *as* $s_0 \overset{p_1}{\leadsto}_{\theta_1} s_1 \cdots s_{n-1} \overset{p_n}{\leadsto}_{\theta_n} s_n$ *such that* $s \equiv s_0$, $t \equiv s_n$, $\theta \equiv \theta_1 \cdots \theta_n$, *and* $p_i \in B_{i-1}$ *for* $1 \leq i \leq n$.

Example 3.3. *Consider the TRS* $\mathcal{R} = \{\mathtt{a} \to \mathtt{0}, \mathtt{f(x)} \to \mathtt{h(x)}\}$ *and input term* $\mathtt{f(a)}$*. The following narrowing derivation is not basic*

$$\underline{\mathtt{f(a)}} \leadsto_{id, f(x)\to h(x)} \mathtt{h(\underline{a})} \leadsto_{id, a \to 0} \mathtt{h(0)}$$

since position 1*, selected in the second narrowing step, is not basic (the narrowing redex* $\mathtt{a}$ *was introduced by instantiation of the rhs* $\mathtt{h(x)}$ *of the second rule). A basic narrowing derivation is* $\mathtt{f(a)} \leadsto_{id,a\to 0} \mathtt{f(0)} \leadsto_{id,f(x)\to h(x)} \mathtt{h(0)}$.

3.1.1 A novel result on termination of basic narrowing

As mentioned above, [*Hullot*, 1981] proved two different results for *basic narrowing*:

1. its unification-completeness for canonical TRSs, and
2. its termination for canonical TRSs where all basic narrowing derivations issuing from the right-hand side of every rule terminate.

We note here that in contrast to ordinary narrowing, termination of basic narrowing does not imply termination of rewriting. If canonicity is replaced by simple confluence in item 2 above, then completeness is lost, even if we restrict ourselves to normalizable substitutions [*Middeldorp and Hamoen*, 1994]. Unification-completeness of basic narrowing can be restored (for normalizable substitutions) by additionally requiring $\mathcal{R}$ to be right-linear [*Middeldorp and Hamoen*, 1994].

The termination of basic narrowing was established in Hullot's Ph.D. thesis for canonical TRSs as follows.

[1] The correct termination result, which only guarantees the termination of basic narrowing under the same assumptions, was established in [*Hullot*, 1981], and subsequently referred to in a number of works [*Hölldobler*, 1989; *Middeldorp and Hamoen*, 1994; *Réty*, 1987].

Proposition 3.4 (Termination of Basic Narrowing for Canonical TRSs [*Hullot*, 1981, Proposition 7.1])**.** *Let $\mathcal{R}$ be a canonical TRS. If for every $l \to r \in \mathcal{R}$, all basic narrowing derivations issuing from r terminate, then any* basic *narrowing derivation in $\mathcal{R}$ issuing from any term terminates.*

Hullot's condition on the rhs's of rewrite rules is essential for the termination of basic narrowing, as illustrated in the following example.

Example 3.5. [*Chabin and Réty*, 1991] *Consider the canonical TRS $\mathcal{R} = \{\mathtt{h}(\mathtt{f}(\mathtt{y})) \to \mathtt{h}(\mathtt{y})\}$. The following infinite basic narrowing derivation can be proved:*

$$\underline{\mathtt{h}(\mathtt{x})} \leadsto_{\{x \mapsto \mathtt{f}(\mathtt{y})\},\mathcal{R}} \underline{\mathtt{h}(\mathtt{y})} \leadsto_{\{y \mapsto \mathtt{f}(\mathtt{y}')\},\mathcal{R}} \underline{\mathtt{h}(\mathtt{y}')} \ldots$$

A termination result similar to Proposition 3.4 does not hold for ordinary narrowing, even when the condition is extended to demand termination of ordinary narrowing for the rhs's of the rules (instead of the less demanding condition of basic narrowing termination). The TRS of Example 1.4 would be an easy counterexample.

By considering a particular class of TRSs where there is a precise correspondence between basic narrowing and ordinary narrowing derivations, we are able to identify the conditions which enable a termination result for ordinary narrowing formulated in Hullot's style.

The class where basic and full narrowing derivations coincide was first identified by Rety in a commutation result for narrowing sequences [*Réty*, 1987]. Réty's commutation result is based on the condition that narrowing produces only normalized substitutions, as formalized in the following definition.

Definition 3.6 (Rety's normalization condition)**.** [*Réty*, 1987] *A TRS $\mathcal{R}$ satisfies Rety's normalization condition if, for every term s, every substitution θ computed by an ordinary narrowing derivation issuing from s satisfies that $\theta\!\upharpoonright_{Var(s)}$ is normalized.*

To continue, we need to introduce the notion of *antecedent* of a position in a rewrite sequence.

Definition 3.7 (Antecedent of a position [*Réty*, 1987])**.** *Let $t \xrightarrow{p}_{l \to r} t'$ be a rewriting step, $v \in \mathcal{P}os(t)$, and $v' \in \mathcal{P}os(t')$. We say position v is an* antecedent *of v' iff*

1. *$v \parallel p$, i.e., v is incomparable to p, and $v \equiv v'$, or*
2. *there is a variable $x \in Var(r)$, $u' \in \mathcal{P}os_x(r)$, and $u \in \mathcal{P}os_x(l)$ s.t. $v' \equiv p.u'.w$ and $v \equiv p.u.w$.*

This notion extends to a rewrite sequence by transitive closure of the rewriting relation in the usual way.

With the notations of the previous definition, we have:

1. $t|_v \equiv t'|_{v'}$,
2. v' may have no antecedent if $v' = p.u'$ with $u' \in \mathcal{P}os_\Sigma(r)$, or if $v' < p$,

3. v' may have several antecedents if l is not linear.

Therefore, the notion of antecedent is (nearly) dual to the standard notion of *descendants* of a position in a rewrite sequence [*TeReSe*, 2003]. The main difference is that, given a rewriting step $t \xrightarrow{p}_{l\to r} t'$ and a position q such that $q \leq p$, then q is not an antecedent of any position in t' whereas the same position q in t' is commonly considered the descendant of q in t.

Definition 3.8 (Terminal antecedents [*Réty*, 1987])**.** *Let $\mathcal{S}$ be a rewrite sequence $t_0 \to_{\mathcal{R}} t_1 ... \to_{\mathcal{R}} t_n$, and $q_n \in \mathcal{P}os(t_n)$. Given an antecedent $q_i \in \mathcal{P}os(t_i)$ of q_n, we say that q_i is* terminal *in $\mathcal{S}$ iff either $i = 0$ or q_i has no antecedent in t_{i-1}.*

The notion of antecedent can be extended to narrowing as follows:

Definition 3.9 (Narrowing antecedent of a position [*Réty*, 1987])**.** *Let $t \leadsto^*_{\sigma,\mathcal{R}} t'$, $v \in \mathcal{P}os(t)$, and $v' \in \mathcal{P}os(t')$. We say v is a (terminal) antecedent of t iff v is a (terminal) antecedent of v' in the rewrite sequence $t\sigma \to^*_{\mathcal{R}} t'$.*

The following proposition holds.

Proposition 3.10 ([*Réty*, 1987])**.** *Given a narrowing sequence*

$$t_0 \overset{p_1}{\leadsto}_{\sigma_1,l_1\to r_1} t_1 t_{n-1} \overset{p_n}{\leadsto}_{\sigma_n,l_n\to r_n} t_n$$

if $q_i \in \mathcal{P}os(t_i)$ is an antecedent of $q_n \in \mathcal{P}os(t_n)$, then $(t_i(\sigma_{i+1}..\sigma_n))|_{q_i} \equiv t_n|_{q_n}$.

As we mentioned, when $\mathcal{R}$ is not left-linear, a given position may have several antecedents in a previous term in the derivation, and may also have antecedents in different previous terms which are not antecedents from one another. Therefore, a position may have terminal antecedents in different previous terms of the sequence.

Also note that, whenever an expression is introduced by instantiation, and subsequently propagated along the narrowing derivation, its terminal antecedents are all in the initial input term of the sequence, and occur exactly at the positions of the input term which become instantiated. This is due to the absence of extra variables in rhs's.

The following commutation property is the key of our proof. For $\vartheta \equiv \vartheta_1 \cdots \vartheta_k$, we use $t \overset{u^1,\ldots,u^k}{\leadsto}_{\vartheta,l\to r} s$ as a shorthand to denote the narrowing sequence $t \overset{u^1}{\leadsto}_{\vartheta_1,l\to r} s_1 \ldots \overset{u^k}{\leadsto}_{\vartheta_k,l\to r} s$.

Proposition 3.11 (Maximum commutation)**.** [*Réty*, 1987] *Let $\mathcal{R}$ be a right-linear TRS, which is also either left-linear or conservative. Consider a narrowing sequence*

$$t_0 \overset{p_1}{\leadsto}_{\sigma_1,l_1\to r_1} t_1 \cdots t_{n-1} \overset{p_n}{\leadsto}_{\sigma_n,l_n\to r_n} t_n$$

such that $\sigma_1 \cdots \sigma_n$, restricted to $Var(t_0)$, is normalized. Then, there exists a commuted *narrowing derivation*

$$\begin{array}{l}
t_0 \overset{u_1^1,\ldots,u_1^{k_1}}{\rightsquigarrow}_{\theta_1, l_n \to r_n} \overset{p_1}{\rightsquigarrow}_{\sigma'_1, l_1 \to r_1} t'_1 \\
\vdots \\
t'_{n-2} \overset{u_{n-1}^1,\ldots,u_{n-1}^{k_{n-1}}}{\rightsquigarrow}_{\theta_{n-1}, l_n \to r_n} \overset{p_{n-1}}{\rightsquigarrow}_{\sigma'_{n-1}, l_{n-1} \to r_{n-1}} t'_{n-1} \\
t'_{n-1} \overset{p_n}{\rightsquigarrow}_{\theta_n, l_n \to r_n} t_n
\end{array}$$

such that $\theta_1\sigma'_1 \cdots \theta_{n-1}\sigma'_{n-1}\theta_n \equiv \sigma_1 \cdots \sigma_n[Var(t_0)]$, where $u_1^1, \ldots, u_i^{k_i}$ are the terminal antecedents of position p_n in term t_i.

The following commutation result for ordinary narrowing derivations easily follows.

Proposition 3.12 (Commutation of narrowing derivations)**.** *Let $\mathcal{R}$ be a TRS that satisfies Rety's normalization condition as well as the conditions for Rety's maximum commutation property (i.e. right-linearity, and either left-linearity or conservativeness). For every narrowing sequence $t_0 \overset{p_1}{\rightsquigarrow}_{\theta_1,\mathcal{R}} t_1 \cdots \overset{p_n}{\rightsquigarrow}_{\theta_n,\mathcal{R}} t_n$, there is a commuted basic narrowing sequence $t_0 \overset{q_1}{\rightsquigarrow}_{\sigma_1,\mathcal{R}} t'_1 \cdots \overset{q_m}{\rightsquigarrow}_{\sigma_m,\mathcal{R}} t'_m$ such that $t'_m \equiv t_n$, and $\theta_1 \cdots \theta_n \equiv \sigma_1 \cdots \sigma_m[Var(t_0)]$.*

Proof. By successive applications of Proposition 3.11.

Given the narrowing sequence $t_0 \overset{p_1}{\rightsquigarrow}_{\theta_1,\mathcal{R}} t_1 \cdots \overset{p_n}{\rightsquigarrow}_{\theta_n,\mathcal{R}} t_n$, assume that p_i is the first non-basic position selected in the derivation. By Proposition 3.11, we can commute the derivation so that the step i is performed on the terminal antecedent positions of p_i. Those terminal antecedents occur at basic positions, since redexes are never introduced in a basic narrowing derivation by instantiation due to the Rety's normalization condition. Note that the procedure that repeatedly applies Proposition 3.11 to the derivation which results from the previous commutation is finite since the number of non-basic steps to commute is reduced at each application. □

Together with the normalization condition, Réty's "maximal commutation property" of narrowing sequences requires two additional conditions: right-linearity, and either left-linearity or conservativeness [*Réty*, 1987]. By requiring all these properties, we are able to achieve the desired narrowing termination result.

Theorem 3.13 (Termination of Narrowing)**.** *Let $\mathcal{R}$ be a right-linear TRS which satisfies Réty's normalization condition and is either left-linear or conservative. If basic narrowing terminates in $\mathcal{R}$, then ordinary narrowing also terminates in $\mathcal{R}$.*

Proof. By contradiction. Assume that there exists an infinite narrowing derivation $\mathcal{D}$ issuing from a given term t. Then, we can obtain infinitely many finite subsequences (prefixes) of $\mathcal{D}$. By Proposition 3.12, each of these finite subsequences has a corresponding, commuted basic narrowing derivation issuing from t. Hence, there are infinitely many basic narrowing derivations issuing from the very same term t, each of which is: (i) finite (by definition), and (ii) a prefix of the subsequent one (by Proposition 3.12), which yields a contradiction. □

A popular class of TRSs that satisfy the normalization condition is the class of left-linear constructor systems [*Reddy*, 1985], that only compute[2] constructor substitutions. Nevertheless, in Section 3.2.1 we are able to define a more general, syntactic characterization of TRSs satisfying this condition.

Note that Example 1.4 satisfies all conditions required in Theorem 3.13, except for Réty's normalization condition. In the following section, we improve this result by explicitly getting rid of canonicity.

3.2 Beyond canonical systems

Hullot's basic narrowing termination result for canonical TRSs, recalled in Proposition 3.4, has been referred to in a number of works, e.g. [*Middeldorp and Hamoen*, 1994; *Hölldobler*, 1989; *Réty*, 1987]. However, to the best of our knowledge, no one has explicitly pointed out that canonicity is not explicitly used in Hullot's proof. This seems to suggest that canonicity of $\mathcal{R}$ might be superfluous for Hullot's basic narrowing termination result and that is only required for deriving both termination and unification completeness of the basic narrowing mechanism in one go. By providing a new proof for Hullot's basic narrowing termination result, in this section we confirm this presumption and demonstrate that canonicity can be safely removed.

The following result establishes the termination of basic narrowing without the canonicity requirement.

Theorem 3.14 (Termination of Basic Narrowing). *Let $\mathcal{R}$ be TRS. If for every $l \to r \in \mathcal{R}$, all basic narrowing derivations issuing from r terminate, then every basic narrowing derivation issuing from any term terminates.*

To prove this result, we find it useful to use the alternative definition of basic narrowing given in [*Hölldobler*, 1989]. In this formulation, elements of the derivation are split into a *skeleton* and an *environment* part. The environment part keeps track of the accumulated substitutions so that, at each step, substitutions are composed in the environment part, but are not applied to the expressions in the skeleton part, as opposed to ordinary narrowing. Due to this representation, the basic occurrences in $t\theta$ are all in t, whereas the non-basic occurrences are all in the codomain of θ. This ensures that no narrowing step will reduce any expression brought by a substitution computed in a previous step.

Definition 3.15 (Basic narrowing (skeleton-environment)[*Hölldobler*, 1989]). *Given a term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ and a substitution σ, a basic narrowing step for $\langle s, \sigma\rangle$ is defined by $\langle s, \sigma\rangle \overset{b}{\leadsto}_{p,\mathcal{R},\theta} \langle t, \sigma'\rangle$ if there exist $p \in \mathcal{P}os_\Sigma(s)$, $l \to r \ll \mathcal{R}$, and substitution θ such that $\theta = mgu(s|_p\sigma, l)$, $t = (s[r]_p)$, and $\sigma' = (\sigma\theta)[t]$.*

In the proof we make use of the following auxiliary lemma relating the basic narrowing derivations starting from the instances of a term.

[2]This is desired in some functional logic languages [*Hanus*, 1994], since a broader class of solutions may contain unevaluated or undefined expressions.

Lemma 3.16. *Let $\mathcal{R}$ be a TRS, t be a term, and σ be a substitution. Let n be the length of the longest basic narrowing derivation for $\langle t, \sigma\rangle$ in $\mathcal{R}$. Then, for every substitution ϑ, n is an upper bound for the length of the basic narrowing derivations issuing from $\langle t, \sigma\vartheta\rangle$ in $\mathcal{R}$.*

Proof. By induction on n.

The case when $n = 0$ is straightforward, since no basic narrowing step issuing from $\langle t, \sigma\vartheta\rangle$ can be proved for any ϑ, either.

Consider now the case when $n > 0$. If there is no basic narrowing sequence such that the substitution θ computed in the first step $\langle t, \sigma\rangle \overset{b}{\leadsto}_{\mathcal{R},\theta} \langle t', \sigma\theta\rangle$ is compatible with ϑ, then there is no basic narrowing sequence issuing from $\langle t, \sigma\vartheta\rangle$, and the conclusion follows. Assume that $\langle t, \sigma\rangle \overset{b}{\leadsto}_{\mathcal{R},\theta} \langle t', \sigma\theta\rangle$ is the first step of a basic narrowing derivation for $\langle t, \sigma\rangle$ such that θ is compatible with ϑ. Since ϑ and θ are compatible, the basic narrowing step $\langle t, (\sigma\vartheta)\rangle \overset{b}{\leadsto}_{\mathcal{R},\theta'} \langle t', (\sigma\vartheta)\theta'\rangle$ can be proven, and $(\sigma\vartheta)\theta'$ is compatible with $\sigma\theta$. By hypothesis, the lengths of the basic narrowing derivations issuing from $\langle t', \sigma\theta\rangle$ are bounded by $n - 1$, hence so are the lengths of the basic narrowing derivations issuing from $\langle t', (\sigma\vartheta)\theta'\rangle$, which concludes the proof. □

The proof of Theorem 3.14 follows now by structural induction with help of the Lemma just introduced.

Proof of Theorem 3.14. We prove the slightly more general result that, for every term t and substitution σ, every basic narrowing derivation issuing from $\langle t, \sigma\rangle$ terminates. We proceed by structural induction on the term t.

- The case when t is a variable is straightforward.
- Let $t \equiv f(t_1, \ldots, t_m)$, $m \geq 0$, and consider any basic narrowing derivation $\mathcal{D}$: $\langle t, \sigma\rangle \overset{b}{\leadsto}_{\mathcal{R},\theta_1,p_1} \langle t_2, \sigma_2\rangle \overset{b}{\leadsto}_{\mathcal{R},\theta_2,p_2} \cdots$ stemming from $\langle t, \sigma\rangle$. We distinguish two cases: either none of the positions p_j for $j > 0$ is ϵ, or there is $k > 0$ such that the k-th basic narrowing step in $\mathcal{D}$ takes place at the root position of t_k. In the first case, by the induction hypothesis the derivation terminates, since every basic narrowing derivation issuing from $\langle t_i, \sigma\rangle$ terminates, for $i \in \{1, \ldots, m\}$. In the second case, $\langle t_k, \sigma_k\rangle \overset{b}{\leadsto}_{\{l\to r\},\theta_k,\epsilon} \langle r, \sigma_{k+1}\rangle$. Since all basic narrowing derivations issuing from r terminate, then by Lemma 3.16 the derivation terminates. Thus, the conclusion follows. □

Note that the termination of basic narrowing in $\mathcal{R}$ does not imply that $\mathcal{R}$ is terminating for rewriting, as shown by the example below.

Example 3.17. *Consider the following non-terminating and non-confluent TRS $\mathcal{R}$ borrowed from [Toyama, 1987], which satisfies Réty's normalization condition[3]:*

$$\mathtt{f(b,c,x)} \to \mathtt{f(x,x,x)} \qquad \mathtt{a} \to \mathtt{b} \qquad \mathtt{a} \to \mathtt{c}$$

By applying Theorem 3.14, there is no infinite basic narrowing derivation in $\mathcal{R}$.

[3]by Theorem 3.26 in Section 3.2.1 ahead.

The following Hullot-like termination result follows from Theorem 3.14.

Corollary 3.18 (Termination of Narrowing)**.** *Let $\mathcal{R}$ be a right-linear TRS which satisfies Réty's normalization condition and is either left-linear or conservative. If for every $l \to r \in \mathcal{R}$ all basic narrowing derivations issuing from r terminate, then every narrowing*[4] *derivation issuing from any term terminates.*

Proof. It follows immediately from Theorem 3.13 and Theorem 3.14. □

Example 3.19. *Consider the following linear TRS $\mathcal{R}$ satisfying*[5] *Réty's normalization condition:*

$$\mathtt{f(a,x)} \to \mathtt{a} \qquad \mathtt{f(f(b,x),a)} \to \mathtt{c(h(x))} \qquad \mathtt{h(c(x))} \to \mathtt{x}$$

By applying Corollary 3.18, since all basic narrowing derivations issuing from the rhs's of the rules in $\mathcal{R}$ terminate, then narrowing terminates in $\mathcal{R}$.

Note that right-linearity is essential for Réty's maximum commutation property and hence cannot be dropped from Corollary 3.18, as shown in the following example.

Example 3.20. *Consider again the TRS of Example 3.17, which also satisfies Réty's normalization condition. However, note that it is not right-linear. Basic narrowing terminates in this TRS, as seen before, but an infinite ordinary narrowing sequence exists for input term* $\mathtt{f(a,a,a)}$*, which is set off when we instantiate the rhs* $\mathtt{f(x,x,x)}$ *of the first rule using the non-normalized binding* $\{x \mapsto \mathtt{a}\}$*:*

$$\mathtt{f(\underline{a},a,a)} \to \mathtt{f(b,\underline{a},a)} \to \underline{\mathtt{f(b,c,a)}} \to \mathtt{f(\underline{a},a,a)} \to \mathtt{f(b,\underline{a},a)} \cdots .$$

Unfortunately, both Hullot's termination condition based on the rhs's of rewrite rules and Réty's normalization condition are not syntactical. Hullot's termination condition has been approximated in the related literature by the following syntactic criterion, assuming that $\mathcal{R}$ terminates: every non-ground rhs of a rewrite rule is a constructor term [*Dershowitz et al.*, 1992; *Prehofer*, 1994]. This generalizes the original characterization given by Hullot [*Hullot*, 1980], who required all non-ground rhs's to be variables. Note that these syntactic characterizations do not work under the conditions of Theorem 3.14 since termination is not explicitly required, and we would require also ground rhs's to be constructor terms (the rule $\mathtt{a} \to \mathtt{a}$ would be an easy counter–example).

With regard to Réty's normalization condition, we already mentioned a popular class of TRSs satisfying this property: left-linear constructor systems. In the following section, we demonstrate that Réty's condition also holds in the more general class of left-linear, `rnf`-based TRSs. This leads to a practical approximation of the termination result for ordinary narrowing given in Corollary 3.18 which holds in (a subclass of) linear, `rnf`-based TRSs.

[4]In the following, narrowing always refers to *full* (unrestricted) narrowing.

[5]It satisfies the sufficient characterization of TRSs satisfying Réty's normalization condition given in Section 3.2.1.

Moreover, by further exploring the notion of rigid normal form, in Sections 3.3 and 3.4, we will be also able to generalize the popular approximation of Hullot's termination condition based on the rhs's of the rules, and provide stronger (purely syntactical in some cases) termination results for ordinary narrowing in a class of systems where right-linearity as well as left-linearity are no longer required.

3.2.1 Rigid normal forms and rnf-based TRSs

Let us define the class of rnf-based TRSs by introducing the notion of *rigid normal form*[6] (rnf), which lifts the standard notion of (rewriting) normal form to narrowing.

Definition 3.21 (Rigid normal form)**.** *A term s is a* rigid normal form *(*rnf*) if there is no term t, substitution θ, and position p such that $s \overset{p}{\rightsquigarrow}_{\theta,\mathcal{R}} t$.*

The notion of rigid normal form is stronger than the standard notion of rewriting normal form but can still be easily decided by simply checking that no subterm of the considered term unifies with the lhs of any rule in $\mathcal{R}$. This notion extends to *rigidly normalized substitutions* in the obvious way.

We define the new class of rnf-based TRSs as follows.

Definition 3.22 (rnf-pattern)**.** *A term $f(t_1, ..., t_n) \in \mathcal{T}(\Sigma, \mathcal{V})$ is a* rnf*-pattern if, for all i s.t. $1 \leq i \leq n$, t_i is a rigid normal form.*

Definition 3.23 (rnf-based TRS)**.** *Given a TRS $\mathcal{R}$, we call it* rnf-based *if the left-hand side of every rule in $\mathcal{R}$ is a* rnf*-pattern.*

Note that two popular classes of rnf-based, left-linear TRSs are: (i) left-linear constructor systems, and (ii) almost orthogonal TRSs, i.e., typical functional programs.

Proposition 3.24 (Almost orthogonal rnf-basedness)**.** *Almost orthogonal TRSs are* rnf*-based.*

Proof. By definition of almost orthogonal TRS, every critical pair is an overlay, i.e., two lhs's overlap only at the root position. Therefore, the lhs of every rewrite rule is a rnf-pattern. □

The following result is instrumental and shows that rigid normal forms are closed under substitution.

Lemma 3.25. *For every rigidly normalized substitution θ, if t is a rigid normal form, then $t\theta$ is also a rigid normal form.*

[6] Our rnf notion is more general than the *strongly $\rightsquigarrow$-irreducible* terms proposed in [*Escobar et al.*, 2006] for topmost theories, where t is strongly $\rightsquigarrow$-irreducible if $t\sigma$ is a normal form for every normalized substitution σ. Consider, e.g. the non-confluent, non-topmost TRS $\mathcal{R} = \{\mathtt{f(a)} \rightarrow \mathtt{b}, \mathtt{a} \rightarrow \mathtt{b}\}$. The term $\mathtt{f(x)}$ is strongly $\rightsquigarrow$-irreducible, since non-normalized substitutions such as $\{\mathtt{x} \mapsto \mathtt{a}\}$ are not considered within the definition. However, it is not a rigid normal form.

Proof. By contradiction. Let us assume that $t\theta$ is not a rigid normal form, i.e., there is a term s, substitution σ, rule R, and $p \in \mathcal{P}os(t\theta)$ such that $t\theta \overset{p}{\leadsto}_{\sigma,R} s$. Actually, since θ is rigidly normalized, then $p \in \mathcal{P}os_{\Sigma}(t)$. Therefore, we have that $t\theta|_p$ and l unify with unifier σ, whereas by the rnf assumption $t|_p$ and l do not unify, which leads to contradiction. □

From Lemma 3.25, it follows that, in rnf-based left-linear TRSs, all substitutions computed by narrowing are rigidly normalized, hence also normalized, and therefore these systems satisfy Rety's normalization condition.

Theorem 3.26 (Rigid normalization)**.** *Let $\mathcal{R}$ be a* rnf*-based, left-linear TRS. Every substitution θ computed by an ordinary narrowing derivation issuing from the term t satisfies that $\theta\restriction_{\mathcal{V}ar(t)}$ is rigidly normalized.*

Proof. Consider a narrowing sequence

$$t \equiv t_0 \overset{p_1}{\leadsto}_{\theta_1, l_1 \to r_1} t_1 \cdots t_{n-1} \overset{p_n}{\leadsto}_{\theta_n, l_n \to r_n} t_n \equiv s$$

At each narrowing step $t \overset{p}{\leadsto}_{\theta, l \to r} s$, the substitution $\theta\restriction_{\mathcal{V}ar(t)}$ is rigidly normalized, since l is linear and every subterm of l is a rigid normal form. We proceed by induction on n. The base case $n = 0$ is trivial. For the case when $n > 0$, by induction hypothesis we have that $\vartheta \equiv (\theta_1 \cdots \theta_{n-1})\restriction_{\mathcal{V}ar(t)}$ is rigidly normalized, i.e., for each binding $x \mapsto w \in \vartheta$, we have that w is a rigid normal form. Now, by Lemma 3.25, we have that $w\theta_n$ is also a rigid normal form, and the conclusion follows. □

From Theorem 3.26 and Corollary 3.18, the following practical criterion for termination of narrowing in rnf-based, linear TRSs easily follows.

Corollary 3.27 (Termination of narrowing as termination of basic narrowing)**.** *Let $\mathcal{R}$ be a linear,* rnf*-based TRS. If for every $l \to r \in \mathcal{R}$, all basic narrowing derivations issuing from r terminate, then every narrowing derivation issuing from any term terminates.*

3.3 Beyond right-linear systems

Our narrowing termination results in Section 3.2.1 rely on Réty's commutation result [*Réty*, 1987], which requires right-linearity and either left-linearity or conservativeness. In this section, we provide new termination results that are not based on Réty's commutation property, and thus get rid of linearity in some cases.

The notions of *root-stable rigid normal form* (rs−rnf) and *stable rigid normal form* (srnf) are the key for achieving termination when right-linearity is dropped.

3.3.1 Stable and Root-stable Rigid normal forms

Let us highlight the insufficiency of considering rigid normal forms for ensuring the narrowing termination when right-linearity of $\mathcal{R}$ is not imposed. Basically, the problem lies in the fact that rigid normal forms are not stable under instantiation by non-normalized substitutions, as illustrated in the following example.

Example 3.28. *Consider again the left-linear and* rnf*-based TRS $\mathcal{R}$ of Example 3.17, which is non-confluent and not right-linear. The term* $\mathtt{f(x,x,x)}$ *in the rhs of the first rule is a rigid normal form since it does not unify with lhs* $\mathtt{f(b,c,x)}$*; hence, it cannot be narrowed. However, the instance* $\mathtt{f(a,a,a)}$ *is no longer a rigid normal form, since it can be rewritten (in two steps) to* $\mathtt{f(b,c,a)}$*, which can then be rewritten (hence narrowed) at the top position by using the first rule of $\mathcal{R}$.*

Let us introduce the notion of root-stable rigid normal form, which lifts to narrowing the standard notion of root-stable (or head) normal form; and the notion of stable rigid normal form, which denotes a variable or a rigid normal form stable under instantiation.

Then, a suitable definition of "stable rigid normal form t" is provided which ensures that every subterm s of t is conveniently "protected", in the sense that no instantiation can enable a "non-topmost" rewriting sequence such that then the resulting term can be narrowed at the top.

Definition 3.29 (stable and root-stable rigid normal forms)**.** *A term s is a* root-stable rigid normal form *(*rs−rnf*) if either s is a variable or there are no substitutions θ and θ' and terms s' and s'' s.t. $s\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} s' \overset{\epsilon}{\leadsto}_{\theta'} s''$. A term t is a* stable rigid normal form *(*srnf*) if every subterm of t is a root-stable rigid normal form.*

The above notions extend to *root-stable rigidly normalized substitutions* and *stable rigidly normalized substitutions* in the natural way.

Note that the notion of stable rigid normal form is stronger than the notion of rigid normal form. Example 3.28 above shows that the inverse does not hold. By definition, non-variable stable rigid normal forms are stable under instantiation, even under non-normalized substitutions. Also, constructor terms as well as ground normal forms are trivial cases of stable rigid normal forms. Therefore, the approximation of Hullot's basic narrowing termination condition based on checking that the rhs's of the rules are constructor terms is subsumed by the more general right-srnf condition.

Definition 3.30 (Right-rnf TRS)**.** *A TRS is called* right-rnf *if the right-hand side of every rule in $\mathcal{R}$ is a rigid normal form. Similarly, a TRS is* right-srnf *if the right-hand side of every rule in $\mathcal{R}$ is a stable rigid normal form.*

The following interesting property holds.

Proposition 3.31 (Termination of right-srnf systems)**.** *Rewriting is terminating in every right-*srnf *TRS.*

Proof. (Sketch) We apply the dependency pairs technique [*Arts and Giesl*, 2000] (cf. Section 4.1). Since by definition a right-srnf TRS $\mathcal{R}$ can have no chains, then $\mathcal{R}$ terminates by [*Arts and Giesl*, 2000, Thm. 6] (cf. Theorem 4.4). □

Note that the right-srnf condition required in Proposition 3.31 cannot be weakened to right-rnf. The TRS of Example 3.17 is an easy counterexample.

In order to provide a general termination result for right-srnf TRSs, we need the following notion.

Definition 3.32 (Stable rigid normalization condition (SRNC)). *A TRS $\mathcal{R}$ satisfies the* stable rigid normalization condition *if, for every term s, every substitution θ computed by an ordinary narrowing derivation issuing from s satisfies that $\theta\lceil_{Var(s)}$ is stable rigidly normalized.*

By requiring the SRNC (instead of Réty's maximal commutation condition), we are able to provide the following termination result for narrowing.

Theorem 3.33 (Termination of narrowing under the SRNC). *Let $\mathcal{R}$ be a right-*`srnf` *TRS that satisfies the stable rigid normalization condition. Every narrowing derivation issuing from any term terminates.*

Proof. The proof for this theorem is subsumed by the proof for Theorem 3.43 in the following section, and thus we omit it here. □

The following example demonstrates that stable rigid normal forms cannot be replaced by vanilla rigid normal forms in Theorem 3.33.

Example 3.34. *Consider again the left-linear and* `rnf`*-based TRS of Example 3.17, where we showed that the term* $\mathtt{f(x,x,x)}$ *in the rhs of the first rule is a rigid normal form. However it is not a* `srnf`*, and narrowing does not terminate for the input term* $\mathtt{f(a,a,a)}$*, as shown in Example 3.17.*

In the following section, we characterize the class of TRSs where all rigid normal forms are stable, thus guaranteeing that the new structure that is introduced through ordinary narrowing steps by instantiation cannot lead to an infinite derivation. This is the final ingredient we need in order to derive a purely syntactical characterization of narrowing termination which does not require the right-linearity of $\mathcal{R}$.

3.3.2 Reachability-complete TRSs

Let us introduce a new class of TRSs (which we call reachability-complete TRSs) where narrowing is strongly reachability-complete. This is inspired by the commonly used terminology which, recalling the unification-completeness of narrowing for canonical TRSs, uses the name "complete TRS" as an alternative terminology to refer to this particular class [*Knuth and Bendix*, 1970; *Hullot*, 1980; *Middeldorp and Hamoen*, 1994].

Definition 3.35 (Reachability-complete TRS). *A TRS $\mathcal{R}$ is reachability-complete iff the narrowing procedure is strongly reachability-complete for $\mathcal{R}$.*

The following interesting result holds for reachability-complete TRSs.

Proposition 3.36 (Reachability complete `srnf` systems). *Let $\mathcal{R}$ be a reachability-complete TRS. If s is a rigid normal form, then s is also a stable rigid normal form.*

Proof. By contradiction. Assume that s is a rigid normal form and there is a position p in s such that $s|_p$ is not a root-stable rigid normal form. Then, there are two substitutions ρ and ρ' and terms t and t' such that $s|_p\rho \overset{>\epsilon}{\to}{}^*_{\mathcal{R}} t \overset{\epsilon}{\leadsto}_{\rho'} t'$. Let $s' = s[t']_p$. Since $\mathcal{R}$ is reachability-complete, for the reachability goal $s \to^* s'$ narrowing computes a solution η more general than $\rho\rho'$ s.t. $s \leadsto^*_\eta s''$, with $s'' \leq s'$. Hence, s is not a rigid normal form, which contradicts the initial assumption. $\square$

Proposition 3.36 reveals that reachability-completeness can be understood as the property that shelters rigid normal forms with a suitable form of stability which suffices to prevent non-normalized bindings from introducing the possibility of initiating an infinite narrowing derivation. Actually, under reachability-completeness we are able to weaken stable rigid normal forms down to the purely syntactic notion of rigid normal form, which is easier to check.

As a corollary of Theorem 3.33, by using Proposition 3.36, we achieve the following termination result for reachability-complete TRSs. Note that reachability-complete TRSs that satisfy Réty's normalization condition also satisfy SRNC.

Corollary 3.37 (Termination of Narrowing under Rety's normalization condition)**.** *Let $\mathcal{R}$ be a reachability-complete, right-*rnf *TRS which satisfies Réty's normalization condition. Every narrowing derivation issuing from any term terminates.*

In the above result, reachability-completeness allows us to get rid of right-linearity, e.g. in TRSs that are confluent or topmost [*Meseguer and Thati*, 2007]. Unfortunately, this is not the case for left-linearity, which is still required in the sufficient criteria for Réty's normalization condition.

Inspired by Christian's narrowing termination result for left-flat TRSs [*Christian*, 1992], in the last section of this chapter we further refine the termination result above by getting rid of left-linearity.

3.4 Beyond left-linear systems

In [*Christian*, 1992], termination of narrowing was proved for left-flat TRSs (i.e., each argument occurring in the lhs of a rewrite rule is either a variable or a ground term), provided the rewrite rules are also compatible with a termination ordering $<$. Given a signature Σ, a termination ordering $<$ is a well-founded monotone ordering on $\mathcal{T}(\Sigma, \mathcal{V})$ which is also stable under substitutions; i.e., if $s < t$, then $s\sigma < t\sigma$ for any substitution σ (see [*Dershowitz*, 1987] for a survey on termination orderings). Christian formalized a "harmlessness" criterion for narrowing as an extension $<_{\mathcal{L}}$ of $<$ as follows: $s <_{\mathcal{L}} t$ whenever the number of distinct variables in s is either (i) less than the number in t; or (ii) equal to the number in t, and s and t are identical everywhere, except at some position p such that $s|_p < t|_p$. Then he demonstrated that, whenever any term t narrows to t' in a left-flat system, then $t' <_{\mathcal{L}} t$, which ensures termination of narrowing.

Informally, the reason why left-flat rules "behave well" is that they do not introduce *new* variables in the term: each narrowing step either reduces the number of distinct variables, or produces a smaller term under the $<$ well-founded ordering.

Example 3.38. *Consider the following non-flat TRS* $\mathtt{f(f(x))} \to \mathtt{f(x)}$ *which can be oriented with the following termination ordering:* $t > s$ *iff* $|t\sigma| > |s\sigma|$ *for every substitution* σ*, where* $|t|$ *denotes the size of* t*. However, this rule raises the infinite narrowing sequence*

$$\underline{\mathtt{f(x)}} \leadsto_{\{\mathtt{x}\mapsto\mathtt{f(x')}\}} \underline{\mathtt{f(x')}} \leadsto_{\{\mathtt{x'}\mapsto\mathtt{f(x'')}\}} \underline{\mathtt{f(x'')}} \leadsto_{\{\mathtt{x''}\mapsto\mathtt{f(x''')}\}} \cdots$$

Note that the ultimate source of narrowing non-termination in this TRS is the introduction of "fresh variables" $\mathtt{x'}$*,* $\mathtt{x''}$*, which causes the terms* $\mathtt{f(x')}$*,* $\mathtt{f(x'')}, \ldots$ *to enter at some point in the derivation, whereas* $\mathtt{f(x')} \not<_{\mathcal{L}} \mathtt{f(x)}$*.*

In order to combine and generalize the termination results that hold for TRSs which are either left-flat [*Christian*, 1992] or rnf-based (Section 3.3), we extend the stable rigid normalization condition (SRNC) as follows. Informally, the key idea is to ensure that the substitutions applied in narrowing steps cannot introduce any new term that is not a root stable rigid normal form and may only replicate in the worst case (strict) subterms of existing ones.

Definition 3.39 (Quasi stable rigidly normalized substitution)**.** *Given a TRS* $\mathcal{R}$*, a term* s*, a substitution* θ *is* quasi stable rigidly normalized w.r.t. s and $\mathcal{R}$ *(QSRN) if, for each variable* $x \in Var(s)$ *that appears in* s *more than once,* $x\theta$ *is either*

(i) a ground term,

(ii) a stable rigid normal form, or

(iii) there exists a position $p \in \mathcal{P}os_{\Sigma}(s)$ *such that* $x\theta \equiv (s\theta)|_p$*.*

Note that every substitution is quasi stable rigidly normalized w.r.t. a linear term, for any TRS.

Example 3.40. *Consider the TRS* $\{\mathtt{f}(\mathtt{f}(x)) \to x\}$ *of of Example 1.4, and the term* $s = \mathtt{c(c(x, f(x)), f(y))}$*. Assume* $\mathtt{a}$ *is a fresh symbol not in the signature of* R*.*
The following substitutions are QSRN w.r.t. s *and* R*:*

- $\{\mathtt{x} \to \mathtt{a}\}$*, by (i)*
- $\{\mathtt{x} \to \mathtt{c(z, z)}\}$*, by (ii)*
- $\{\mathtt{x} \to \mathtt{f(y)}\}$*, by (iii).*

Note that $\{\mathtt{x} \to \mathtt{f(z)}\}$ *is not QSRN w.r.t.* s *and* R *because it is neither ground, a* srnf*, or a subterm of* s*.*

The following result is trivial due to linearity.

Corollary 3.41. *In a right-linear TRS* $\mathcal{R}$*, every substitution computed by narrowing for a linear term* s *is quasi stable rigidly normalized w.r.t.* s *and* $\mathcal{R}$*.*

Definition 3.42 (Quasi stable rigid normalization condition (QSRNC))**.** *A TRS $\mathcal{R}$ satisfies the* quasi stable rigid normalization condition *if, for every term s, every substitution θ computed by an ordinary narrowing derivation issuing from s satisfies that $\theta\restriction_{Var(s)}$ is quasi stable rigidly normalized w.r.t. s and $\mathcal{R}$.*

Note that SRNC implies QSRNC. Now we are ready to provide our most general result for narrowing termination.

Theorem 3.43 (Termination of narrowing under QSRNC)**.** *Let $\mathcal{R}$ be a right-*srnf *TRS that satisfies the quasi stable rigid normalization condition. Every narrowing derivation issuing from any term terminates.*

Before proceeding with the proof of Theorem 3.43, we need to introduce a number of auxiliary definitions and results. A term that is not a root-stable rigid normal form is called a non-rs$-$rnf. First, let us recall the standard notion of multiset.

Definition 3.44 (Multiset [Baader and Nipkow, 1998])**.** *A multiset M over a set A is a function $M : A \to \mathcal{N}$. Intuitively, $M(x)$ is the number of copies of $x \in A$ in the multiset M.*

Next, we define a pair of measure functions D and D^* over terms which produce the multiset of non root-stable rigid normal subterms of a term.

Definition 3.45 (Multiset of non root-stable rigid normal forms)**.** *Let $\mathcal{R}$ be a TRS and t be a term. We define $D^*_{\mathcal{R}}(t)$ (resp. $D_{\mathcal{R}}(t)$) to be the multiset of subterms (resp. non-ground subterms) of t that are not root-stable rigid normal forms.*

We drop the subindex $\mathcal{R}$ in $D_{\mathcal{R}}(t)$ and $D^*_{\mathcal{R}}(t)$ when it is clear from the context.

Example 3.46. *Assume any TRS $\mathcal{R}$ such that any term rooted by symbol f is not a root-stable rigid normal form w.r.t. $\mathcal{R}$, whereas terms rooted by symbols a or s are root-stable rigid normal forms. Then,*

1. *for $t_1 = f(a,a)$, we have $D(t_1) = \emptyset$ and $D^*(t_1) = \{f(a,a)\}$,*
2. *for $t_2 = f(s(x), f(a,a))$, we have $D(t_2) = \{f(s(x), f(a,a))\}$ and $D^*(t_2) = \{f(s(x), f(a,a)), f(a,a)\}$,*
3. *for $t_3 = f(f(x,y),a)$, we have $D(t_3) = D^*(t_3) = \{f(f(x,y),a), f(x,y)\}$,*
4. *for $t_4 = f(f(x,y), f(x,y))$, we have $D(t_4) = D^*(t_4) = \{f(f(x,y), f(x,y)), f(x,y), f(x,y)\}$, and*
5. *for $t_5 = f(f(x,y), f(x',y'))$, we have $D(t_5) = D^*(t_5) = \{f(f(x,y), f(x',y')), f(x,y), f(x',y')\}$.*

Let us now define an ordering $\rhd_\theta$ on terms. The main idea behind the definition is to capture that whenever t narrows to t' with substitution θ, all non-rs$-$rnf terms in t' are just descendants of (possibly instantiated) strict subterms of non-rs$-$rnf terms of t.

Definition 3.47 ($\triangleright_\theta$ and $\blacktriangleright_\theta$)**.** *Let t, s be two terms and θ a substitution. We say $t \triangleright_\theta s$ if there is a position $p \in \mathcal{P}os_\Sigma(t)$ such that $s \equiv t\theta|_p$ and either $p > \epsilon$ or $\theta \neq id$. We write $t \blacktriangleright_\theta s$ whenever $t \triangleright_\theta s$ and s is a strict subterm of t (i.e., $p > \epsilon$).*

We recall the definition of a multiset ordering.

Definition 3.48 (Multiset ordering)**.** [Baader and Nipkow, 1998] *Let $(M, \succ)$ be a partial ordering. The extension of $\succ$ to multi-sets is defined by the two axioms below, where M_1 and M_2 are multisets over a set S:*

$$M_1 \succ_{mul} M_2 \Leftrightarrow M_1 \not\equiv M_2 \tag{3.1}$$

$$\forall m \in S, M_2(m) > M_1(m) \Rightarrow \exists m' \in S : (m' \succ m, M_1(m') > M_2(m')) \tag{3.2}$$

.

Since a term might be instantiated further and further, the orderings $\triangleright_\theta$ and $\blacktriangleright_\theta$ are not well-founded, hence neither of their multiset extensions $\triangleright_\theta^{mul}$ and $\blacktriangleright_\theta^{mul}$ are well-founded. Nevertheless, we can prove that there are no infinite decreasing sequences generated by narrowing steps. Informally, the idea is that no new non-rs−rnf terms are introduced by narrowing and it may replicate in the worst case (strict) subterms of existing ones.

Definition 3.49 (Non-additive)**.** *We say a decreasing sequence $S_0 \triangleright_{\theta_1}^{mul} S_1 \triangleright_{\theta_2}^{mul} \cdots \triangleright_{\theta_n}^{mul} S_n$ of term multisets is* non-additive *if no new terms are introduced at any step of the sequence, i.e., for every $i > 0$ and term t in S_i such that $S_i(t) > 0$, there is a term t' in S_{i-1} such that $S_{i-1}(t') > 0$ and $t \equiv t'\theta_i$.*

Definition 3.50 (Monotonically decreasing)**.** *We say a non-additive decreasing sequence $S_0 \triangleright_{\theta_1}^{mul} S_1 \triangleright_{\theta_2}^{mul} \cdots \triangleright_{\theta_n}^{mul} S_n$ of term multisets is* monotonically decreasing *if replication of a term t implies consumption of a term u lying strictly above t, i.e., for every $i > 0$ and terms t in S_i and t' in S_{i-1} such that $t \equiv t'\theta_i$, $S_{i-1}(t') > 0$, and $S_i(t) > S_{i-1}(t')$, there are terms u in S_i and u' in S_{i-1} such that $u \equiv u'\theta_i$, $S_{i-1}(u') > S_i(u)$, and $u' \blacktriangleright_{id} t'$.*

Lemma 3.51. *Every monotonically decreasing sequence of term multisets is finite.*

Proof. By contradiction. Let us assume an infinite monotonically decreasing sequence

$$S_0 \triangleright_{\theta_1}^{mul} S_1 \triangleright_{\theta_2}^{mul} \cdots \triangleright_{\theta_n}^{mul} S_n \cdots$$

Since it is non-additive, there must be a term u_0 in the original multiset S_0 that is replicated infinitely many times, i.e., for all i there is u_i in S_i such that, for some p, $u_i \equiv u_0\theta_1 \cdots \theta_i|_p$ and $S_i(u_i) \geq S_0(u_0)$. However, this leads to a contradiction since the sequence is monotonically decreasing and u_0 is finite. □

We prove that that the conditions of the previous result do hold for the class of TRSs that we are considering.

Lemma 3.52. *Let $\mathcal{R}$ be a right-*srnf *TRS that satisfies the quasi stable rigid normalization condition. For each narrowing sequence* $t_0 \overset{p_1}{\leadsto}_{\theta_1,l_1\to r_1} t_1 \cdots t_{n-1} \overset{p_n}{\leadsto}_{\theta_n,l_n\to r_n} t_n \cdots$ *the sequence* $D(t_0) \rhd_{\theta_1}^{mul} D(t_1) \rhd_{\theta_2}^{mul} D(t_2) \cdots$ *of term multisets is monotonically-decreasing.*

Proof. The proof that the sequence is non-additive is obtained by considering that new non-ground, non-rs$-$rnf terms are never introduced by narrowing steps, since (i) $\mathcal{R}$ is right-srnf, and (ii) the computed substitutions are QSRNC and thus any eventual new non-rs$-$rnf brought by instantiation is ground.

The proof that the sequence is monotonically decreasing is obtained by considering that any new non-rs$-$rnf term u of t_i is ground, and any non-rs$-$rnf subterm u of t_{i-1} that has more occurrences in t_i than in t_{i-1} satisfies $t_{i-1}|_{p_i} \blacktriangleright_{id} u$. $\square$

Two further auxiliary results follow: (i) for the case when a narrowing step produces a stable rigidly normalized substitution, and (ii) for the case when a narrowing step produces a quasi stable rigidly normalized substitution. Intuitively, when a term t narrows to t', we take into account the number of variables of t and t' and the number of non-rs$-$rnf subterms in t and t', and show that at least one of these numbers decreases.

We first prove that whenever a term t narrows to t' by computing a stable rigidly normalized computed substitution θ, $D^*(t) \rhd_{\theta}^{mul} D^*(t')$.

Lemma 3.53. *Let $\mathcal{R}$ be a right-*srnf *TRS. For every narrowing step* $t \overset{p}{\leadsto}_{\theta,l\to r} t'$ *such that θ is a stable rigidly normalized substitution,* $D^*(t) \rhd_{\theta}^{mul} D^*(t')$.

Proof. By Definition 3.48, let us assume that there exists a term u such that $D^*(t')(u) > D^*(t)(u)$; otherwise it is trivial. We have to prove that there is a subterm w of t s.t. $w \rhd_\theta u$ and $D^*(t)(w) > D^*(t')(w)$. We consider the cases when $D^*(t)(u) = 0$ and $D^*(t)(u) > 0$ separately.

If $D^*(t)(u) = 0$, then u does not appear in t because u is an instantiated version of a subterm u' of t. That is, since θ is a stable rigidly normalized substitution and r is a stable rigid normal form, there is a subterm u' of t such that $u \equiv u'\theta$ and $\theta\lceil_{Var(u')} \not\equiv id$. Therefore, $u' \rhd_\theta u$, $D^*(t)(u') > D^*(t')(u') = 0$, and the conclusion follows.

If $D^*(t)(u) > 0$, then the extra occurrences of u in t' have been introduced by propagation of the applied substitution due to the possible non-linearity of r (the possible non-linearity of l did not have any effect because θ is stable rigidly normalized), which implies that u is a strict subterm of $t|_p$. However, we have that $D^*(t)(t|_p) > D^*(t')(t|_p)$ (at least in one unit since $t|_p$ has been narrowed) and $t|_p \rhd_\theta u$, since u is a subterm of $t|_p$. Therefore, the conclusion follows. $\square$

The previous result can be easily extended to $D(t)$ instead of $D^*(t)$ when we consider narrowing steps on non-ground terms.

Lemma 3.54. *Let $\mathcal{R}$ be a right-*srnf *TRS. For every narrowing step* $t \overset{p}{\leadsto}_{\theta,l\to r} t'$ *such that $t|_p$ is non-ground and θ is a stable rigidly normalized substitution,* $D(t) \rhd_{\theta}^{mul} D(t')$.

Proof. Immediate. $\square$

Now we are ready to extend the previous results to the case when the computed substitutions are only quasi stable rigidly normalized.

Lemma 3.55. *Let $\mathcal{R}$ be a right-*srnf *TRS. For every narrowing step $t \overset{p}{\leadsto}_{\theta,l\to r} t'$ such that $t|_p$ is non-ground and θ is a quasi stable rigidly normalized substitution w.r.t. t, $D(t) \ \triangleright_\theta^{mul} \ D(t')$.*

Proof. By Definition 3.48, let us assume that there exists a non-ground term u such that $D(t')(u) > D(t)(u)$; otherwise it is trivial. We have to prove that there is a subterm w of t s.t. $w \triangleright_\theta u$ and $D(t)(w) > D(t')(w)$. We consider the cases when $D(t)(u) = 0$ and $D(t)(u) > 0$ separately.

If $D(t)(u) = 0$, then, since θ is a quasi stable rigidly normalized substitution w.r.t. t and r is a stable rigid normal form, there is a subterm u' of t such that $u \equiv u'\theta$ and $\theta\lceil_{Var(u')} \not\equiv id$. Therefore, $u' \triangleright_\theta u$, $D(t)(u') > D(t')(u') = 0$, and the conclusion follows.

If $D(t)(u) > 0$, then the extra occurrences of u in t' have been introduced by propagation of the applied substitution, due to the possible non-linearity of either l or r. In both cases, u is a strict subterm of $t|_p$, and since $t|_p$ is non-ground and r is a stable rigid normal form, $D(t)(t|_p) > D(t')(t|_p)$ (at least in one unit), $t|_p \blacktriangleright_\theta u$, and the conclusion follows. $\square$

And finally, let us proceed with the proof of Theorem 3.43.

Proof of Theorem 3.43. Given a narrowing sequence

$$\mathcal{D} = t_0 \overset{p_1}{\leadsto}_{\theta_1,l_1\to r_1} t_1 \cdots t_{n-1} \overset{p_n}{\leadsto}_{\theta_n,l_n\to r_n} t_n \cdots$$

we define an order based on pairs $\langle D(t_i), D^*(t_i)\rangle$ and ordered by $\langle M_1, M_2\rangle \succ_\theta \langle M_1', M_2'\rangle$ if $M_1 \ \triangleright_\theta^{mul} \ M_1'$ or $M_1 = M_1'$ and $M_2 \ \triangleright_\theta^{mul} \ M_2$. Note that the order is noetherian due to Lemmas 3.51 and 3.52. Then, we prove termination of narrowing by noetherian induction on $\langle D(t_n), D^*(t_n)\rangle$ and $\succ_{\theta_n}$.

1. (Base case) $\langle D(t_n), D^*(t_n)\rangle = \langle \emptyset, \emptyset\rangle$, which implies that there are no narrowable subterms in t_n, and the claim follows trivially.

2. (Induction case) We have $\langle D(t_n), D^*(t_n)\rangle \neq \langle \emptyset, \emptyset\rangle$, and consider the subsequent narrowing step
$$t_n \overset{p_{n+1}}{\leadsto}_{\theta_{n+1},l_{n+1}\to r_{n+1}} t_{n+1}$$
We consider the following three cases separately,

   (a) if $t_n|_{p_{n+1}}$ is a ground term, then $D(t_n) = D(t_{n+1})$ and θ_{n+1} is a stable rigidly normalized substitution. Then by Lemma 3.53, $D^*(t_n) \ \triangleright_{\theta_{n+1}}^{mul} \ D^*(t_{n+1})$;

   (b) if $t_n|_{p_{n+1}}$ is a non-ground term and θ_{n+1} is a stable rigidly normalized substitution, then by Lemma 3.54, $D(t_n) \ \triangleright_{\theta_{n+1}}^{mul} \ D(t_{n+1})$;

(c) if $t_n|_{p_{n+1}}$ is a non-ground term and θ_{n+1} is a quasi stable rigidly normalized substitution w.r.t. t_n, then by Lemma 3.55, $D(t_n) \rhd^{mul}_{\theta_{n+1}} D(t_{n+1})$.

In the three cases, the result follows by induction hypothesis. □

Theorem 3.43 and Corollary 3.41 together provide the following result.

Corollary 3.56 (Termination of Narrowing for right-linear TRSs)**.** *Let $\mathcal{R}$ be a right-linear, right-*srnf *TRS. Every narrowing derivation in $\mathcal{R}$ issuing from any linear term terminates.*

Now we are ready to introduce the notion of *left-plain* TRSs as a natural generalization, with regard to narrowing termination, of both left-flat as well as rnf-based TRSs. Note that the case of a variable argument is considered in the definition below, since variables are rigid normal forms.

Definition 3.57 (Left-plain TRS)**.** *A TRS $\mathcal{R}$ is called* left-plain *if every non-ground strict subterm of the left-hand side of every rule of $\mathcal{R}$ is a rigid normal form.*

Example 3.58. *The following TRS defining a specialized version of the* `xor` *operator used in many security protocols [Comon-Lundh, 2004; Cortier et al., 2006] is left-plain. The symbol* `h` *is constructor; it might represent e.g. the hash of a message.*

$$\mathtt{x}+\mathtt{x} \to \mathtt{0} \qquad \mathtt{x}+\mathtt{0} \to \mathtt{x} \qquad (\mathtt{0}+\mathtt{0})+\mathtt{h}(\mathtt{x}) \to \mathtt{h}(\mathtt{x})$$

Note that the third rule is neither left-flat nor `rnf`*-based.*

Example 3.59. *The rule $\mathtt{0}+(\mathtt{0}+\mathtt{x}) \to \mathtt{x}$ is not left-plain, since the non-ground subterm $\mathtt{0}+\mathtt{x}$ is not a rigid normal form. Indeed, the following infinite narrowing derivation can be proved*

$$\mathtt{c}(\underline{\mathtt{0}+\mathtt{x}},\mathtt{x}) \leadsto_{\{x\mapsto \mathtt{0}+\mathtt{x}'\}} \mathtt{c}(\mathtt{x}',\underline{\mathtt{0}+\mathtt{x}'}) \leadsto_{\{x'\mapsto \mathtt{0}+\mathtt{x}''\}} \mathtt{c}(\underline{\mathtt{0}+\mathtt{x}''},\mathtt{x}'')\cdots$$

By using Proposition 3.36, we are able to demonstrate the QSRNC property for left-plain, reachability-complete TRSs.

Lemma 3.60. *Every left-plain, reachability-complete TRS satisfies the quasi stable rigid normalization condition.*

Proof. By reachability-completeness, we can safely consider rigid normal forms instead of stable rigid normal forms. On the other hand, since the composition of two rigidly normalized substitutions is also rigidly normalized, we can safely consider the substitutions computed at each narrowing step.

Let us consider a term t and the narrowing step $t \overset{p}{\leadsto}_{\sigma,l\to r} t'$. We prove the result by induction on the number of bindings in σ. If $\sigma = id$, the conclusion follows straightforwardly. Let $x \mapsto u \in \sigma$ and suppose that u does not satisfy any of the conditions (i), (ii), and (iii) of Definition 3.39, i.e., u is not ground, is not a rigid normal form, and is not a non-variable subterm of $t\sigma$. By definition, there is at least one position $p' \in \mathcal{P}os(l) \cap \mathcal{P}os(t|_p)$ s.t. $t|_{p.p'} = x$ and $t\sigma|_{p.p'} = l\sigma|_{p'} = u$. Let us consider an arbitrary such p'. We distinguish the cases when $l|_{p'}$ is a variable or not. If $l|_{p'} = y \in \mathcal{V}$, then $y \mapsto u \in \sigma$ and y must be a repeated variable in l, since u is not a variable (it is not a rigid normal form) and σ is the most general unifier. Therefore, there is a position $p'' \in \mathcal{P}os_\Sigma(t|_p)$ s.t. $t\sigma|_{p.p''} = u$. But this contradicts condition (iii) of Definition 3.39. If $l|_{p'} \notin \mathcal{V}$, then $l|_{p'}$ itself is not ground and is not a rigid normal form, since x cannot appear in $l|_{p'}$ and, by induction hypothesis, $\sigma \setminus \{x \mapsto u\}$ satisfies conditions (i), (ii), and (iii) of Definition 3.39. However, this contradicts condition (ii) of Definition 3.39, and the conclusion follows. □

By using Lemma 3.60, the following result directly follows as a specialization of Theorem 3.43 for left-plain TRSs.

Corollary 3.61 (Termination of Narrowing for left-plain TRSs)**.** *Let $\mathcal{R}$ be a left-plain, reachability complete, right-*`rnf` *TRS. Every narrowing derivation issuing from any term terminates.*

Note that the above result is very handy as it can be applied to TRSs which are neither purely left-flat or `rnf`-based, as illustrated in Example 3.58.

Finally, by using the known results for the strong reachability-completeness of narrowing given by [*Meseguer and Thati*, 2007], we are able to particularize Corollary 3.61 to a number of purely syntactical, non-trivial classes of TRSs where narrowing has a finite search space and is still (strongly) complete as a procedure to solve reachability goals. The following result also subsumes Corollary 3.56.

Corollary 3.62 (Termination of Narrowing for right-`rnf` TRSs)**.** *Let $\mathcal{R}$ be a right-*`rnf` *TRS which is either*

1. *right-linear,*
2. *confluent and left-plain, or*
3. *topmost.*

Then, every narrowing derivation issuing from any term terminates. In the case of (1), the termination (proved in Corollary 3.56) only holds for linear input terms.

Example 3.63. *Let us consider the following rule defining the exponentiation function used as a primitive operation for key exchange in the Diffie-Hellman key agreement protocol [Comon-Lundh, 2004; Cortier et al., 2006], where symbols $*$ and* `g` *are constructors*[7]*.*

$$\mathtt{exp}(\mathtt{exp}(\mathtt{g}, \mathtt{y}), \mathtt{z}) \to \mathtt{exp}(\mathtt{g}, \mathtt{y} * \mathtt{z})$$

This rule satisfies both criteria 1 and 2 of Corollary 3.62, hence we conclude that narrowing derivations w.r.t. this rule terminate.

The criteria given in Corollary 3.62 are particularly practical, since many interesting TRSs fit in one of the above classes. For instance, termination of the following TRSs follows from Corollary 3.62 straightforwardly (other examples are given in Table 3.1):

[7] $*$ is commonly defined as a (built-in) associative commutative operator with identity element `1`.

- almost orthogonal, right-rnf TRSs (including right-rnf orthogonal TRSs as a particular case);
- constructor, confluent, and right-rnf TRSs;
- right-linear, right-rnf TRSs (only for linear input terms).

Note that the TRS in Example 1.1 satisfies all the above requirements, except for the condition to be right-rnf.

We would like to note that our results are not comparable to those of [*Christian*, 1992], i.e., we do not claim to subsume Christian's results. As a counterexample, it suffices to consider any left-flat TRS that is compatible with a termination ordering but is neither right-rnf nor reachability-complete. Obviously, [*Christian*, 1992] does not subsume our results either, since Christian's criterion cannot deal with TRSs that are not left-flat.

The main advantage of our approach w.r.t. [*Christian*, 1992] is that our criteria are truly syntactic and do not rely on termination orderings. As an additional advantage, note that some of our results are based (and hence preserve) the strong reachability-completeness of $\mathcal{R}$, besides ensuring the narrowing termination, which is not guaranteed by Christian's result.

3.5 Discussion

It should be pointed out that, even if these results may seem of reduced interest for programming languages since the right-rnf condition precludes recursion, they are still very relevant for proving the termination of narrowing-based procedures that are used in the context of bottom-up program analysis and abstract diagnosis. As a representative example, the compact collecting semantics of term rewriting systems defined on top of narrowing in [*Alpuente et al.*, 2010a] gives rise to interpretations (systems of semantic equations) which satisfy the right-srnf condition. For instance, the rules defining insertion at the end of a linked list:

$$\begin{aligned}\texttt{insert}(Y, \texttt{cons}(X, XX)) &\to \texttt{cons}(X, \texttt{insert}(Y, XX))\\ \texttt{insert}(Y, Z) &\to \texttt{cons}(Y, Z)\end{aligned}$$

give rise to an interpretation, computed by means of narrowing-based procedures, of the form:

$$\begin{aligned}\texttt{insert}(X, Z) &\to \texttt{cons}(X, Z)\\ \texttt{insert}(X, \texttt{cons}(Y, Z)) &\to \texttt{cons}(Y, \texttt{cons}(X, Z))\\ \texttt{insert}(X, \texttt{cons}(Y1, \texttt{cons}(Y2, Z))) &\to \texttt{cons}(Y1, \texttt{cons}(Y2, \texttt{cons}(X, Z)))\\ &\dots\end{aligned}$$

where the right hand sides of the equations are srnf. The proof of termination of narrowing over these interpretations is critical to ensure the computability of the semantics.

It is challenging to identify more general classes of TRSs where narrowing terminates without giving up the ability to test (almost purely) syntactic properties of individual rewrite rules. Let us emphasize that all the results in this section also apply to proving termination of sophisticated narrowing strategies such as innermost or lazy narrowing [*Hanus*, 1994], where narrowing steps are restricted to a suitable subset of the term positions. Obviously, more general classes of TRSs may exist where a particular narrowing strategy terminates.

Theorem 3.13 provides a powerful criterion for proving narrowing termination in TRSs or theories where basic narrowing terminates, often called BNT-theories; see e.g. [*Schmidt-Schauß*, 1988]. Moreover, under the conditions for Theorem 3.13, the modularity results of Chapter 7 apply also to full narrowing. On the other hand, [*Nieuwenhuis*, 1996] demonstrated that, for some kinds of theories closed under some basic inference rules, equational unification can be proved terminating by again applying these inference rules. This entails termination of basic narrowing e.g. in shallow theories (where all variables in the axiomatization are shallow) that are saturated under a rule which subsumes basic narrowing, called basic paramodulation. A similar result holds in standard theories, which extend shallow theories by only requiring shallowness to the variables that appear on both sides of the equations.

4 Automatic Termination

In Chapter 3 we identified a number of classes of systems where narrowing terminates. But as mentioned at the conclusion, it is difficult to expand these results while staying in the realm of syntactic restrictions.

In this chapter we introduce a different, non-syntactic approach to proving termination of narrowing, which has the nice property of being highly amenable to automation. The method is based on the well-known approach of Dependency Pairs [*Arts and Giesl*, 2000] for termination of rewriting. In recent years, the dependency pair (DP) method for automating the termination proofs of term rewriting has achieved tremendous success, as witnessed by the large number of publications and tools since its introduction in [*Arts and Giesl*, 2000] and subsequent reformulation in [*Giesl et al.*, 2005c] (see [*Giesl et al.*, 2006b; *Hirokawa and Middeldorp*, 2004] for extensive references thereof). In [*Nguyen et al.*, 2008], the notions of dependency pairs and dependency graphs, which were originally developed for term rewriting, were adapted to the logic programming domain, leading to automated termination analyses that are directly applicable to any definite logic program.

Recently, two techniques for the termination of narrowing have been proposed which are based on the dependency pair approach. In [*Nishida and Miura*, 2006; *Nishida et al.*, 2003], the approach of [*Arts and Giesl*, 2000] is adapted to the termination of narrowing, whereas [*Vidal*, 2008; *Nishida and Vidal*, 2010] proves termination of narrowing w.r.t. a given set of initial queries, relying on a restricted form of termination of rewriting based on relative termination.

In order to be as useful as possible, automatic methods for deciding and proving the termination of narrowing must be applicable to a class of TRSs as large as possible. However, both [*Nishida and Miura*, 2006; *Nishida et al.*, 2003] and [*Vidal*, 2008; *Nishida and Vidal*, 2010] apply only to restricted classes of TRSs, namely right–linear TRSs (i.e., no repeated variables occur in the right–hand sides of the rules), or constructor systems (the arguments of the lhs's of the rules are constructor terms). These two classes are overly restrictive for many of the applications of narrowing mentioned in Section 1.

In this chapter, we present an approach which is able to relax these restrictions and is applicable to any class of TRSs, while at the same time being at least as powerful as the existing approaches.

$$
\begin{array}{lcl}
\mathtt{filter}(\mathtt{pckt}(src, dst, \mathtt{established})) & \to & \mathtt{accept} \\
\mathtt{filter}(\mathtt{pckt}(\mathtt{eth0}, dst, \mathtt{new})) & \to & \mathtt{accept} \\
\mathtt{filter}(\mathtt{pckt}(\mathsf{194.179.1}.x{:}port, dst, \mathtt{new})) & \to & \mathtt{filter}(\mathtt{pckt}(\mathtt{secure}, dst, \mathtt{new})) \\
\mathtt{filter}(\mathtt{pckt}(\mathsf{158.42}.x.y{:}port, dst, \mathtt{new})) & \to & \mathtt{filter}(\mathtt{pckt}(\mathtt{secure}, dst, \mathtt{new})) \\
\mathtt{filter}(\mathtt{pckt}(\mathtt{secure}, dst{:}\ \mathsf{80}, \mathtt{new})) & \to & \mathtt{accept} \\
\mathtt{filter}(\mathtt{pckt}(\mathtt{secure}, dst{:}\ \mathtt{other}, \mathtt{new})) & \to & \mathtt{drop} \\
\mathtt{filter}(\mathtt{pckt}(\mathtt{ppp0}, dst, \mathtt{new})) & \to & \mathtt{drop} \\
\mathtt{filter}(\mathtt{pckt}(\mathsf{123.123.1.1}\ {:}port, dst, \mathtt{new})) & \to & \mathtt{accept} \\
\mathtt{pckt}(\mathsf{10.1.1.1}\ {:}port, \mathtt{ppp0}, s) & \to & \mathtt{pckt}(\mathsf{123.23.1.1}\ {:}port, \mathtt{ppp0}, s) \\
\mathtt{pckt}(\mathsf{10.1.1.2}\ {:}port, \mathtt{ppp0}, s) & \to & \mathtt{pckt}(\mathsf{123.23.1.1}\ {:}port, \mathtt{ppp0}, s) \\
\multicolumn{3}{l}{\mathtt{pckt}(src, \mathsf{123.123.1.1}\ {:}port, \mathtt{new}) \to \mathtt{natroute}(\mathtt{pckt}(src, \mathsf{10.1.1.1}\ {:}port, \mathtt{established}),} \\
 & & \mathtt{pckt}(src, \mathsf{10.1.1.2}\ {:}port, \mathtt{established})) \\
\mathtt{natroute}(a, b) & \to & a \\
\mathtt{natroute}(a, b) & \to & b
\end{array}
$$

Figure 4.1: The $\mathcal{R}_{Policy}$ TRS

Example 4.1. *Consider our running example adapted from [Kirchner et al., 2008], the non–right–linear, non–constructor–based, non–confluent TRS shown in Figure 4.1. This TRS models a security, filtering and routing policy that allows packets coming from external networks to be analyzed. We do not describe the intended meaning of each symbol since it is not relevant, but note the kind of expressivity that is assumed in the domain of rule–based policy specification.*

Narrowing is terminating for this TRS, but it cannot be proved by using any of the existing methods [Nishida and Miura, 2006; Nishida et al., 2003; Vidal, 2008] as it does not fit in the right–linear restriction (due to the third rule for `pckt`*) or the constructor discipline (due to the rules for* `filter`*). It doesn't either fit in any of the syntactic characterizations where narrowing terminates defined in Chapter 3. In the rest of this chapter, we develop techniques that allow us to prove its narrowing termination automatically.*

Structure of the chapter

In Section 4.1 we recall the basic notions of the dependency pair approach from the literature. Then, in Sections 4.2 we dissect the structure of minimal infinite narrowing derivations. By doing that, in Section 4.2.2 we distill a notion of narrowing dependency pair which accurately models them, providing a sound and complete termination criterion for narrowing. Then, in Section 4.2.4 we show how to apply this criterion to demonstrate termination of narrowing by means of termination of rewriting. Section 4.3 introduces the narrowing dependency pair (NDP) framework, which enables the automation of the approach and the integration with other termination techniques. Finally, Section 4.4 concludes.

Part of the results in this chapter have been published in [*Alpuente et al.*, 2008a]. In this chapter we additionally:

- improve on the original definition of narrowing dependency pairs. The new definition is complete, while the original definition was not.
- extend the method to consider TRSs with extra variables (cf. Section 4.2),
- establish a formal connection with the results in Chapter 3 (cf. Section 4.2).
- give an exact characterization of the class of systems where narrowing enjoys the TRAT property (cf. Section 4.2.3),
- consider a notion of usable rules for narrowing (cf. Section 4.3), and
- provide two new NDP processors which enable the decomposition of the narrowing dependency graph in strongly connected components (cf. Theorems 4.47 and 4.52).

4.1 The Dependency Pair Method

The dependency pair technique [*Arts and Giesl*, 2000] is one of the most powerful methods for automated analysis of termination of rewriting systems. Throughout this thesis we consider two extensions of this framework. Firstly, in this chapter it is extended to prove termination of narrowing, by introducing the notion of narrowing dependency pair and narrowing chain. Secondly, in Chapter 5 it is extended twice to 1) consider only derivations that issue from a set of initial terms, and 2) to prove relative termination of a system $\mathcal{R}$ with regard to another system $\mathcal{E}$. In order to make this thesis self-contained, in this section we give a very brief introduction to the basic notions underlying the dependency pair technique. For an extensive reference check [*Giesl et al.*, 2006b].

The dependency pair technique focuses on the dependency relations between defined function symbols, paying particular attention to loops, which are modelled as strongly connected components within a graph of functional dependencies. This dependency graph is extracted from the function calls in the right hand sides of the rules. The notion of dependency pair captures this idea by extracting, from every rule, the set of relevant function calls. Given a TRS $\mathcal{R}$ over a signature Σ, for each $\mathsf{f}/n \in \Sigma$, we let $\mathsf{f}^\sharp/n$ be a new fresh symbol; we often write F (i.e. in uppercase characters) instead of $\mathsf{f}^\sharp$ in the examples. Given a term $\mathsf{f}(t_1,\ldots,t_n)$ with f a defined symbol, we let $t^\sharp$ denote the marked term $\mathsf{f}^\sharp(t_1,\ldots,t_n)$.

Definition 4.2 (Dependency Pair [*Arts and Giesl*, 2000])**.** *Given a TRS $\mathcal{R}$ over a signature $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$, the associated set of dependency pairs, $DP(\mathcal{R})$, is defined as follows:*[1]

$$DP(\mathcal{R}) = \{l^\sharp \to t^\sharp \mid l \to r \in \mathcal{R},\ r \rhd t,\ \textit{and } \mathsf{root}(t) \in \mathcal{D}\}$$

It turns out that every $\to_\mathcal{R}$ derivation can be modelled as a topmost $\to_\mathcal{P}$ derivation modulo the rules of $\mathcal{R}$. This is captured by the notion of chain.

[1]Note that if $\mathcal{R}$ is a TRS, so is $DP(\mathcal{R})$.

Definition 4.3 (Chain)**.** [Arts and Giesl, 2000; Giesl et al., 2006b] *Let $\mathcal{P}, \mathcal{R}$ be two TRSs. A (posibly infinite) sequence of pairs $s_1 \to t_1, s_2 \to t_2, \dots$ from $\mathcal{P}$ is a $(\mathcal{P}, \mathcal{R})$-chain if there exists a substitution σ with $t_i\sigma \to_{\mathcal{R}} s_{i+1}\sigma$ for all i. A chain is said to be minimal if and only if all $t_i\sigma$ are terminating w.r.t. $\to_{\mathcal{R}}$.*

Equivalently, every $(DP(\mathcal{R}), \mathcal{R})$ chain can be mapped to a pure $\to_{\mathcal{R}}$ derivation. This means that proving termination of $\mathcal{R}$ is equivalent to proving the absence of infinite $(DP(\mathcal{R}), \mathcal{R})$-chains. This is a simpler problem, since chains have more structure than $\mathcal{R}$ derivations. Ultimately, it boils down to proving relative termination of $DP(\mathcal{R})$ with regard to $\mathcal{R}$ where, moreover, $DP(\mathcal{R})$ is a topmost system (this viewpoint is developed for instance in [Endrullis et al., 2008]).

The following theorem formalizes the study of termination of rewriting with dependency pairs.

Theorem 4.4 (Termination Criterion [Giesl et al., 2005c])**.** *Let $\mathcal{R}$ be a TRS. The following three properties are equivalent.*

1. *$\to_{\mathcal{R}}$ is terminating.*

2. *there is no infinite $(DP(\mathcal{R}), \mathcal{R})$-chain.*

3. *there is no infinite minimal $(DP(\mathcal{R}), \mathcal{R})$-chain.*

A TRS $\mathcal{R}$ is terminating if and only if no infinite minimal $(DP(\mathcal{R}), \mathcal{R})$–chain exists.

The dependency pairs of a TRS $\mathcal{R}$ can be laid out in a *dependency* graph, where the set of nodes is $DP(\mathcal{R})$ and there is an edge between a pair $s \to t$ and a pair $u \to v$ if an instance of u is reachable from an instance of t.

Definition 4.5 (Dependency Graph [Arts and Giesl, 2000])**.** *Given a TRS $\mathcal{R}$ and a set of pairs $\mathcal{P}$, the dependency graph is the directed graph where the nodes are the elements of $\mathcal{P}$, and there is an edge from $s \to t$ to $u \to v$ if $s \to t$, $u \to v$ is a $(\mathcal{P}, \mathcal{R})$-chain.*

Every chain corresponds to a path in the dependency graph. Since only *infinite* chains can lead to non-termination, it suffices to focus on the cycles of the dependency graph. As shown in [Giesl et al., 2002], a termination problem can be decomposed into smaller problems, one for every cycle in the graph, such that the solution of all the subproblems implies the termination of the original problem. In fact, instead of focusing on *every* cycle, one can focus on every strongly connected component in the graph. This modular decomposition is one of the key strengths of the approach.

We note that computing the dependency graph of a TRS is an undecidable problem, since for any two dependency pairs $s \to t$ and $u \to v$, it is in general undecidable whether they form a chain, i.e. whether there exists a substitution σ such that $t\sigma \to^*_{\mathcal{R}} u$. Several estimations exist in the literature, we again refer the reader to e.g. [Giesl et al., 2006b; Thiemann, 2007] for more information on this subject.

Reduction Pair

To see why 2) and 3) are simpler problems than 1) in the termination criterion of Theorem 4.4, recall that in the classical approach to 1) (see [*Baader and Nipkow*, 1998] for a detailed view), the goal is to find a monotone, stable, well-founded order $>$ on terms such that every rule is strictly decreasing, i.e. $l > r$ for every rule $l \to r \in \mathcal{R}$.

When proving the absence of infinite chains, while the goal is still finding an ordering which orients the rules, the requirements are more relaxed thanks to the additional structure in the problem, since:

- the order $\succ$ used to orient the pairs need not be monotone since the $DP(\mathcal{R})$ system is topmost, and
- the order $\succeq$ used to orient the rules need not be well-founded since only relative termination is required.

Thus we arrive at the notion of reduction pair.

Definition 4.6 (Reduction Pair [*Giesl et al.*, 2005c])**.** *A reduction pair $(\succeq, \succ)$ is a pair of orderings such that:*

1. *$\succeq$ is reflexive, transitive, monotonic and stable under substitutions.*
2. *$\succ$ is a stable well-founded order.*
3. *$\succeq$ and $\succ$ are compatible,i.e., $(\succeq \circ \succ \circ \succeq) \subseteq \succ$.*

By means of reduction pairs one can show the absence of infinite chains, by searching for an order which orients all the rules and pairs. In fact this can be done modularly, see [*Giesl et al.*, 2005c] for the details.

The Dependency Pair framework

The DP framework formalism [*Giesl et al.*, 2005c] is a refinement of the dependency pair technique created with the goal of enabling the modular composition of different termination techniques. The DP framework improves the dependency pair technique in several ways. It is fundamentally more powerful, it is also more applicable thanks to the possibility to combine with other termination techniques, and it is more amenable to automation and implementation.

Definition 4.7 (DP problems and processors [*Giesl et al.*, 2005c])**.** *A DP problem is a tuple $\langle \mathcal{P}, \mathcal{R}, f \rangle$ consisting of two TRSs $\mathcal{R}$ and $\mathcal{P}$ and a minimality flag $f \in \{\mathbf{m}, \mathbf{a}\}$, where $\mathbf{m}$ and $\mathbf{a}$ stand for minimal and arbitrary respectively. We say that a DP problem $\langle \mathcal{P}, \mathcal{R}, f \rangle$ is finite if there is no associated infinite (and minimal if f is $\mathbf{m}$) $(\mathcal{P}, \mathcal{R})$ chain. A DP problem $\langle \mathcal{P}, \mathcal{R}, f \rangle$ is infinite either if it is not finite or if $\to_{\mathcal{R}}$ is not terminating.*

A DP processor *is a function Proc which takes a DP problem and either returns a new set of DP problems or fails. Proc is sound if for any DP problem $\mathcal{M}$, $\mathcal{M}$ is finite whenever all DP problems in $Proc(\mathcal{M})$ are finite.*

Following [*Giesl et al.*, 2005c], we construct a tree whose root is labeled with the problem $\langle DP(\mathcal{R}), \mathcal{R}, \mathbf{m}\rangle$ and whose nodes are produced by application of sound DP processors. $\mathcal{R}$ is terminating if all the leaf nodes of this tree are finite.

There are a large number of DP processors available in the literature, as termination using the DP framework is a very active research topic. For an extensive reference about the most commonly used processors, we refer the reader to [*Thiemann*, 2007].

4.2 Termination of Narrowing with Dependency Pairs

In this section we study the shape of minimal infinite narrowing derivations, arriving to the notion of *echoing* terms and to a classification of such infinite derivations. This, in turn, will enable us to define a notion of narrowing dependency pairs in next section which faithfully captures the behaviour of these infinite derivations.

We start by pointing out that in contrast to rewriting, the narrowing relation is not monotone: $t \rightsquigarrow_{\sigma,\mathcal{R}} s$ does not entail $C[t] \rightsquigarrow_{\sigma,\mathcal{R}} C[s]$ but $C[t] \rightsquigarrow_{\sigma,\mathcal{R}} (C\sigma)[s]$ instead. Recall Example 1.4 from Chapter 3.

Example 4.8 (Example 1.4)**.** *Consider the TRS $\mathcal{R}_1 = \{\mathtt{f}(\mathtt{f}(x)) \to x\}$, and the non–linear term $\mathtt{c}(\mathtt{f}(x), x)$. Then there does not exist an infinite $(\rightsquigarrow_{\mathcal{R}})$-derivation for the subterms $\mathtt{f}(x)$ and x, whereas there is an infinite $(\rightsquigarrow_{\mathcal{R}})$-derivation stemming from $\mathtt{c}(\mathtt{f}(x), x)$ which never performs a narrowing step at the root:*

$$\mathtt{c}(\underline{\mathtt{f}(x)}, x) \rightsquigarrow_{\{x \mapsto \mathtt{f}(x')\}} \mathtt{c}(x', \underline{\mathtt{f}(x')}) \rightsquigarrow_{\{x' \mapsto \mathtt{f}(x'')\}} \mathtt{c}(\underline{\mathtt{f}(x'')}, x'') \ldots$$

As shown by the above example, in the presence of non-linearity the non-monotonicity of narrowing has undesirable effects for its termination, since *narrexes* can be brought into the context by the substitution computed at the preceding narrowing step, thus causing other terms in the context to grow. This *echoing* effect plays a fatal role in the (non–) termination of narrowing.

In all existing previous efforts on termination of narrowing using dependency pairs, the position taken is to banish non-monotonicity altogether and demand that narrowing behaves as a monotone relation. For instance, the method of [*Nishida et al.*, 2003] is restricted to systems that have the so-called *Top Reduced Almost Terminating* (TRAT) property.

Definition 4.9 (TRAT [*Nishida et al.*, 2003])**.** *Given a property P on terms, a term t is said to be a* minimal P term *if t satisfies P but none of the proper subterms of t does. Given a TRS $\mathcal{R}$ and a binary relation $\Rightarrow$ (being $\to_{\mathcal{R}}$ or $\rightsquigarrow_{\mathcal{R}}$), an infinite derivation $t \Rightarrow t_1 \Rightarrow t_2 \ldots$ is called* almost terminating *if t is a minimal non–terminating term w.r.t. $\Rightarrow$. An almost terminating derivation $t \Rightarrow t_1 \Rightarrow t_2 \ldots$ is called* top reduced *if it contains a derivation step at the root position.*

We say that $\Rightarrow$ has the TRAT property if, for every non-terminating term t, there exists a top reduced almost-terminating sequence stemming from one subterm of t.

In [*Nishida et al.*, 2003] it is proved that every monotone relation has the TRAT property. Since the rewriting relation is monotone (i.e., $t \to_{\mathcal{R}} s$ implies $C[t] \to_{\mathcal{R}} C[s]$), then it has the TRAT property for every TRS $\mathcal{R}$. In term rewriting this ensures that, in every almost terminating, infinite derivation, a rewriting step is given at the root (cf. [*Hirokawa and Middeldorp*, 2004, Lemma 1]). There are some classes of TRSs in which narrowing exhibits a monotone or monotone–like behaviour and thus enjoys the TRAT property. [*Nishida et al.*, 2003] considers two such classes: right–linear systems when the initial term of the derivation is linear, and constructor systems (for any initial goal).

But we want to remark here that neither TRAT nor monotonicity are a *necessary* condition for the termination of narrowing, as the following examples shows.

Example 4.10. *Consider the non–constructor TRS $\mathcal{R}_2$ defined by the rules below*

$$\begin{aligned} \mathtt{f}(\mathtt{g}(x)) &\to x \\ \mathtt{g}(x) &\to x \end{aligned}$$

together with the non linear initial term $\mathtt{c}(\mathtt{f}(x), x)$. The only possible derivation for this term (or any other) is finite, whereas the TRS does not fit in any of the syntactic characterizations for TRAT[2], and cannot be proven terminating by any of the existing automated approaches ([Nishida and Miura, 2006; Vidal, 2008; Nishida and Vidal, 2010]), nor does it fit in any of the terminating classes introduced in Chapter 3. However as we will prove in the following, the system is terminating for narrowing.

4.2.1 The *echoing* problem

In the following we switch our attention to generalized TRSs (GTRSs), i.e. TRSs with extra variables. Extra variables are very natural in the context of narrowing, they behave as free or logic variables which get instantiated by narrowing only when necessary. Since our technique is flexible enough to consider this extension, there is no particular reason to restrict ourselves to TRSs with no extra variables.

In rewriting (and narrowing), if a TRS is not terminating then there must be a minimal non-terminating term t such that every subterm of t is terminating. Following [*Hirokawa and Middeldorp*, 2004], let us denote the set of all minimal non-terminating narrowing terms by $\mathcal{T}^{\infty}_{\leadsto}$.

Definition 4.11 ($\mathcal{T}^{\infty}_{\leadsto}$)**.** *Let $\mathcal{R}$ be a GTRS. A term t belongs to the set of minimal non-terminating narrowing terms of $\mathcal{R}$, denoted by $\mathcal{T}^{\infty}_{\leadsto_{\mathcal{R}}}$, if and only if:*

1. *There is an infinite $\leadsto_{\mathcal{R}}$ derivation starting from t,*
2. *There is no infinite $\leadsto_{\mathcal{R}}$ derivation starting from a proper subterm of t.*

[2]Strictly speaking, narrowing enjoys the TRAT property in $\mathcal{R}_2$ —and in any ($\leadsto$)-terminating TRS for that matter— because, since there are no infinite derivations, by vacuity every infinite derivation is top reduced. But this does not help us, since we are trying to *decide* termination.

In rewriting [Hirokawa and Middeldorp, 2004, Corollary 1], such a minimal non-terminating term is rooted by a defined symbol. Crucially, this is not true for narrowing, as shown before in Example 4.8. In that example we had a derivation where a $(\rightsquigarrow_{\mathcal{R}_1})$-terminating term, $\mathtt{f}(x)$, leads to nontermination due to an *echoing*-like behaviour. This is formalized by the notion of echoing term.

Definition 4.12 (Echoing terms)**.** *Let $\mathcal{R}$ be a GTRS. We define the set of* echoing terms *w.r.t. $\mathcal{R}$, denoted by $\mathcal{T}^{\circlearrowleft}_{\mathcal{R}}$, as follows: $s \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}}$ if*

- *there is no infinite $\rightsquigarrow_{\mathcal{R}}$ derivation starting from s,*
- *for any context with two holes C, there is a variable $x \in Var(s)$ such that $C[s, x] \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}}}$.*

Example 4.13. *Consider the TRS $\mathcal{R}_1 = \{\mathtt{f}(\mathtt{f}(x)) \to x\}$ of Example 4.8. We have that $\mathtt{c}(\mathtt{f}(x), x) \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}_1}}$, $\mathtt{f}(x) \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}_1}$, and $\mathtt{f}(\mathtt{c}(\mathtt{f}(x), x)) \notin \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}_1}}$ since it is not minimal.*

Echoing terms are terms which are terminating by themselves, but when placed in a non linear context they can become non-terminating. Echoing terms are the key to characterize minimal infinite narrowing derivations.

Informally, we can classify infinite narrowing derivations starting from a minimal non–terminating term in three categories TOP, ECHOING and HYBRID.

The TOP case is the usual one shared by rewriting and narrowing; there are a number of steps below the root and eventually there is a root step introducing a new minimal non-terminating subterm.

The other two cases are due to non–monotonicity and thus unique to narrowing. Both work by introducing a narrex into the context, and hence both need a context and a non linear term to manifest themselves. In the ECHOING case, the narrowing of an echoing subterm introduces into the context a new echoing subterm that enables the process again, as in Example 4.8.

In the HYBRID case, the reduction of an echoing subterm introduces into the context a minimal non–terminating narrex that spawns an infinite narrowing derivation, as in Example 4.14 below.

Example 4.14. *Consider the following TRS $\mathcal{R}_3$:*

$$\mathtt{f}(\mathtt{g}(x)) \to \mathtt{a} \qquad\qquad \mathtt{g}(x) \to \mathtt{g}(x)$$

$\mathtt{g}(x)$ is in $\mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}_3}}$, i.e., it is a minimal non-terminating term. $\mathtt{f}(x) \notin \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}_3}}$, since only the derivation $\mathtt{f}(x) \rightsquigarrow_{\{x \mapsto \mathtt{g}(x')\}} \mathtt{a}$ can be proven. However, given a fresh symbol $\mathtt{c}$, there is a HYBRID *infinite narrowing derivation stemming from the term $\mathtt{c}(\mathtt{f}(x), x) \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}_3}}$. Therefore, $\mathtt{f}(x) \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}_3}$.*

A further insight comes from observing the source of the echoing narrexes in the ECHOING and HYBRID cases. Suppose we split the substitution σ computed by a narrowing step $t \rightsquigarrow_{l \to r, \sigma} s$ into two pieces, $\sigma \equiv \sigma\restriction_l \uplus \sigma\restriction_t$. The $\sigma\restriction_l$ part of the substitution has the usual effect of propagating narrexes to the right-hand side of the rule. On the other hand, the $\sigma\restriction_t$ part is responsible for the echoing of *narrexes* to the context that can fire a new narrowing step. The narrexes propagated into the context come from the subterms of the left-hand side of the rule, as in the TRS $\mathcal{R}_{Policy}$ of Example 4.1, or from the term being narrowed itself, e.g. when $\mathtt{c}(z, \mathtt{h}(\mathtt{g}(x), z))$ is narrowed to $\mathtt{c}(\mathtt{g}(x), 0)$ by using the rule $\mathtt{h}(y, y) \to 0$ and most general unifier $\{z \mapsto \mathtt{g}(x), y \mapsto \mathtt{g}(x)\}$. When analyzing derivations starting from minimal non-terminating terms, the subterms from the term being narrowed are always $(\rightsquigarrow_{\mathcal{R}})$-terminating. This leads to the conclusion that only bindings from $\sigma\!\upharpoonright_l$ can lead to non-termination in these cases.

The lemma below formalizes the shape of infinite minimal narrowing derivations.

Lemma 4.15. *Let $\mathcal{R}$ be a GTRS. For every term $t \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}}}$, we have that either*

1. (TOP) *there exists a rewrite rule $l \to r \in \mathcal{R}$, substitutions σ, ρ, a term t', and a non-variable subterm u of r such that $t \overset{>\epsilon *}{\rightsquigarrow}_{\rho,\mathcal{R}} t' \overset{\epsilon}{\rightsquigarrow}_{\sigma,l\to r} r\sigma \trianglerighteq u$ and $u \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}}}$;*
2. (HYBRID) *there exists a rewrite rule $l \to r \in \mathcal{R}$, terms t', t'', u, l', substitutions ρ, σ, a position $p > \epsilon$, and a variable x such that $t \overset{>\epsilon *}{\rightsquigarrow}_{\rho,\mathcal{R}} t' \overset{p}{\rightsquigarrow}_{\sigma,l\to r} t''$, $x \in \mathcal{V}ar(t'|_p)$ and $x \in \mathcal{V}ar(t'[\Box]_p)$, $x\sigma \trianglerighteq u$, $u = l'\sigma$, $l' \triangleleft l$, $t'|_p \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}}$, and $u, l' \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}}}$;*
3. (ECHOING) *there exists a rewrite rule $l \to r \in \mathcal{R}$, terms t', t'', l', substitutions ρ, σ, a position $p > \epsilon$, and a variable x such that $t \overset{>\epsilon *}{\rightsquigarrow}_{\rho,\mathcal{R}} t' \overset{p}{\rightsquigarrow}_{\sigma,l\to r} t''$, $x \in \mathcal{V}ar(t'|_p)$ and $x \in \mathcal{V}ar(t'[\Box]_p)$, $x\sigma = l'\sigma$, $l' \triangleleft l$, and $t'|_p, x\sigma \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}}$.*

Proof. Let D be an infinite narrowing sequence stemming from t. Since all proper subterms of t are $(\rightsquigarrow_{\mathcal{R}})$-terminating, D must contain a narrowing step at the root position or there is a narrowing step which computes a substitution that carries a narrex into the context which leads to non-termination. Summarizing, we have $t \overset{>\epsilon *}{\rightsquigarrow}_{\rho,\mathcal{R}} t' \overset{p}{\rightsquigarrow}_{\sigma,l\to r} t''$ and we can distinguish two cases depending on whether there is a root step.

- There is at least one root step, i.e. let $p = \epsilon$ above be the first such root step. We are in the TOP case. By the minimality assumption, all proper subterms of t' are $(\rightsquigarrow_{\mathcal{R}})$-terminating and thus terms brought in by the substitution $\sigma = mgu(t', l)$ are $(\rightsquigarrow_{\mathcal{R}})$-terminating. As $r\sigma$ is not $(\rightsquigarrow_{\mathcal{R}})$-terminating, there is some position q s.t. $u = (r\sigma)|_q$ is not $(\rightsquigarrow_{\mathcal{R}})$-terminating. Moreover $q \in \mathcal{P}os(r)$, because all terms brought in by σ are $(\rightsquigarrow_{\mathcal{R}})$-terminating.
- There is no root step in the entire infinite derivation. Then, there must be a step in which the substitution computed introduces a narrex into the context above p. Observe that this requires that t' is non-linear and then t' can be written as $C[t'|_p, x]$, where x is the duplicated variable introducing a narrex. By definition, $t'|_p$ is an echoing term, and we distinguish two cases depending on whether the narrex $x\theta$ is $(\rightsquigarrow_{\mathcal{R}})$-terminating:
  - The narrex introduced is not $(\rightsquigarrow_{\mathcal{R}})$-terminating, hence it contains a minimal non-terminating term u, i.e. $t'' \triangleright u \in \mathcal{T}^{\infty}_{\rightsquigarrow_{\mathcal{R}}}$. We are in the HYBRID

case. Since all the proper subterms of t are $(\leadsto_{\mathcal{R}})$-terminating, u must be an instance of a proper subterm of l introduced by a binding of σ, i.e. there is a variable $x \in \mathcal{V}ar(t'|_p)$ with $x\sigma \trianglerighteq u$, and a term $u' \triangleleft l$, such that $u = u'\sigma$ and $u' \in \mathcal{T}^{\infty}_{\leadsto_{\mathcal{R}}}$.

- The narrex introduced is $(\leadsto_{\mathcal{R}})$-terminating. We are in the ECHOING case. Since $t'|_p$ is an echoing term, there are only a finite number of $\leadsto_{\mathcal{R}}$ steps which can be given from $t'|_p$. Eventually, a $\leadsto_{\mathcal{R}}$ must be given such that $x\sigma = u$ is a new echoing term. In addition, u must be an instance of a proper subterm of l (to see why, recall that in the definition of echoing term one cannot assume anything about the context surrounding the echoing term).

  Hence, there is some variable $x \in \mathcal{V}ar(t'|_p)$ with $x\sigma \in \mathcal{T}^{\circlearrowleft}_{\mathcal{R}}$. □

As a consequence from Lemma 4.15, it is obvious that for all term $t \in \mathcal{T}^{\infty}_{\leadsto_{\mathcal{R}}}$, either t has a defined root symbol or t is non-linear and contains at least an echoing proper subterm. Moreover, every minimal echoing subterm is rooted by a defined symbol.

Corollary 4.16. *Every term $t \in \mathcal{T}^{\infty}_{\leadsto_{\mathcal{R}}}$ either has a defined root symbol or is non-linear and contains at least one echoing subterm.*

Moreover, every term $t \in \mathcal{T}^{\circlearrowleft}$ has a defined root symbol.

4.2.2 Narrowing Dependency Pairs

In this section we develop the notions of narrowing dependency pair and narrowing chain, and provide a sound and complete criterion for the termination of narrowing that is based on analyzing narrowing chains. We have seen in the previous section that the substitution computed in a narrowing step propagates narrexes into the context, that originate either from (subterms of) the left-hand side or the term being narrowed. Although the narrexes coming from proper subterms of the term being narrowed may lead to non–termination, standard (rewriting) termination analyses already cope with them. On the other hand, narrexes coming from proper subterms of the left-hand sides of the rules are specific to narrowing, and this is what drives the generalization of the notion of dependency pair to narrowing.

To construct the set of dependency pairs, we not only relate the left-hand side of each rule with the root–defined subterms occurring in the corresponding right-hand side, as in standard rewriting DP, but also with its *own* root–defined subterms, i.e., those terms whose root symbol is a defined function. The resulting set of dependency pairs faithfully captures the behaviour of narrowing derivations where narrexes coming from a left-hand side are propagated into the context. As a result, the set of narrowing dependency pairs of a TRS is a superset of its set of rewriting dependency pairs. For instance, the TRS $\mathcal{R}_1$ of Example 4.8 has no rewriting dependency pairs (and hence it is trivially terminating) but it has one narrowing dependency pair, as we will see in the following.

The following definition formalizes this idea by extending the standard notion with a novel kind of dependency pairs, which we call *ll–dependency pairs.*

$$
\begin{array}{lll}
(1) & \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{194.179.1.}x\mathtt{:}p, dst, \mathtt{new})) & \rightarrow \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{secure}, dst, \mathtt{new})) \\
(2) & \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{194.179.1.}x\mathtt{:}p, dst, \mathtt{new})) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{secure}, dst, \mathtt{new}) \\
(3) & \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{158.42.}x\mathtt{.}y\mathtt{:}p, dst, \mathtt{new})) & \rightarrow \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{secure}, dst, \mathtt{new})) \\
(4) & \mathtt{filter}^{\#}(\mathtt{pckt}(\mathtt{158.42.}x\mathtt{.}y\mathtt{:}p, dst, \mathtt{new})) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{secure}, dst, \mathtt{new}) \\
(5) & \mathtt{pckt}^{\#}(\mathtt{10.1.1.1}\ \mathtt{:}p, \mathtt{ppp0}, s) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{123.23.1.1}\ \mathtt{:}p, \mathtt{ppp0}, s) \\
(6) & \mathtt{pckt}^{\#}(\mathtt{10.1.1.2}\ \mathtt{:}p, \mathtt{ppp0}, s) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{123.23.1.1}\ \mathtt{:}p, \mathtt{ppp0}, s) \\
(7) & \mathtt{pckt}^{\#}(src, \mathtt{123.123.1.1;}p, \mathtt{new}) \rightarrow \mathtt{natroute}^{\#} & (\mathtt{pckt}(src, \mathtt{10.1.1.1}\ \mathtt{:}p, \mathtt{established}), \\
 & & \ \mathtt{pckt}(src, \mathtt{10.1.1.2}\ \mathtt{:}p, \mathtt{established})) \\
(8) & \mathtt{pckt}^{\#}(src, \mathtt{123.123.1.1}\ \mathtt{:}p, \mathtt{new}) & \rightarrow \mathtt{pckt}^{\#}(src, \mathtt{10.1.1.1}\ \mathtt{:}p, \mathtt{established}) \\
(9) & \mathtt{pckt}^{\#}(src, \mathtt{123.123.1.1}\ \mathtt{:}p, \mathtt{new}) & \rightarrow \mathtt{pckt}^{\#}(src, \mathtt{10.1.1.2}\ \mathtt{:}p, \mathtt{established}) \\
(10) & \mathtt{filter}^{\#}(\bullet(src, dst)) & \rightarrow \mathtt{pckt}^{\#}(src, dst, \mathtt{established}) \\
(11) & \mathtt{filter}^{\#}(\bullet(dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{eth0}, dst, \mathtt{new}) \\
(12) & \mathtt{filter}^{\#}(\bullet(x, p, dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{194.179.1.}x\mathtt{:}p, dst, \mathtt{new}) \\
(13) & \mathtt{filter}^{\#}(\bullet(x, y, p, dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{158.42.}x\mathtt{.}y\mathtt{:}p, dst, \mathtt{new}) \\
(14) & \mathtt{filter}^{\#}(\bullet(dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{secure}, dst\mathtt{:}\ \mathtt{80}, \mathtt{new}) \\
(15) & \mathtt{filter}^{\#}(\bullet(dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{secure}, dst\mathtt{:}\ \mathtt{other}, \mathtt{new}) \\
(16) & \mathtt{filter}^{\#}(\bullet(dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{ppp0}, dst, \mathtt{new}) \\
(17) & \mathtt{filter}^{\#}(\bullet(p, dst)) & \rightarrow \mathtt{pckt}^{\#}(\mathtt{123.123.1.1}\ \mathtt{:}p, dst, \mathtt{new})
\end{array}
$$

Figure 4.2: Dependency pairs of $\mathcal{R}_{Policy}$

Definition 4.17 (Narrowing Dependency Pair)**.** *Given a GTRS $\mathcal{R}$, we have two types of narrowing dependency pairs:*

- *a* lr–dependency pair *(or* standard *DP) of $\mathcal{R}$ is a pair $l^{\#} \rightarrow t^{\#}$ where $l \rightarrow r \in \mathcal{R}$, $r \trianglerighteq t$, and $root(t) \in \mathcal{D}$.*
- *a* ll–dependency pair *(ll-DP) of $\mathcal{R}$ is a pair $l^{\#}[\bullet(Var(l|_p))]_p \rightarrow l|_p^{\#}$ where $l \rightarrow r \in \mathcal{R}$, $root(l|_p) \in \mathcal{D}$.*

where $\bullet$ is a new variadic[3] *symbol not present in the signature of $\mathcal{R}$. The set of all narrowing dependency pairs of $\mathcal{R}$ is denoted by $NDP_{\mathcal{R}}$.*

Example 4.18. *The TRS $\mathcal{R}_1 = \{\mathtt{f}(\mathtt{f}(x)) \rightarrow x\}$ of Example 4.8 has no lr–dependency pairs and the single ll–dependency pair $\mathtt{F}(\bullet(x)) \rightarrow \mathtt{F}(x)$.*

Example 4.19. *For the TRS $\mathcal{R}_{Policy}$ of Example 4.1 we obtain the narrowing dependency pairs shown in Figure 4.2.*

Recall that our purpose is to prove that there are no infinite narrowing derivations in terms of infinite narrowing *chains*. For narrowing we consider suitable the following definition of chain. As in [Giesl *et al.*, 2006b], we assume that different occurrences of dependency pairs are variable disjoint.

Definition 4.20 (Narrowing Chain)**.** *Let $\mathcal{P}, \mathcal{R}$ be two GTRSs. A possibly infinite sequence of narrowing dependency pairs*

$$s_1 \rightarrow t_1, s_2 \rightarrow t_2, \ldots \in \mathcal{P}$$

is called a ($\mathcal{P}$,$\mathcal{R}$)-narrowing chain if there are terms $u_1, \ldots, u_n$ and substitutions $\sigma_1, \ldots, \sigma_n$, such that:

$$u_1 \overset{\epsilon}{\leadsto}_{\sigma_1, s_1 \rightarrow t_1} t_1\sigma_1 \overset{>\epsilon *}{\leadsto}_{\tau_1, \mathcal{R}} u_2 \overset{\epsilon}{\leadsto}_{\sigma_2, s_2 \rightarrow t_2} t_2\sigma_2 \overset{>\epsilon *}{\leadsto}_{\tau_2, \mathcal{R}} u_3 \overset{\epsilon}{\leadsto}_{\sigma_3, s_3 \rightarrow t_3} t_3\sigma_3 \ldots$$

[3] i.e. used with several different arities.

Moreover, $\mathcal{P}$ is a minimal chain if and only if all $t_i\sigma_i$ are $(\leadsto_{\mathcal{R}})$-terminating.

We often omit the $(\mathcal{P},\mathcal{R})$ prefix when referring to narrowing chains when it is clear from the context.

Example 4.21. *Consider the TRS $\mathcal{R}_1$ and the ll-DP $\mathtt{F}(\bullet(x)) \to \mathtt{F}(x)$ from Example 4.18. There is an infinite narrowing chain $\mathtt{F}(\bullet(x)) \to \mathtt{F}(x), \mathtt{F}(\bullet(x)) \to \mathtt{F}(x), \ldots$, as witnessed by the following $\leadsto_{\{\mathtt{F}(\bullet(x))\to\mathtt{F}(x)\}\cup\mathcal{R}_1}$ derivation:*

$$\mathtt{F}(\bullet(x)) \leadsto_{\{\}} \mathtt{F}(x) \leadsto_{\{x\mapsto\bullet(x')\}} \mathtt{F}(x') \leadsto_{\{x'\mapsto\bullet(x'')\}} \mathtt{F}(x'') \cdots$$

In the following we will see how every chain can be mapped to a $\leadsto_{\mathcal{R}}$ derivation. Hence an infinite chain constitutes an infinite $\leadsto_{\mathcal{R}}$ derivation, and proving termination amounts to proving the absence of infinite chains.

Let us briefly mention that the original definition of narrowing dependency pair introduced in [*Alpuente et al.*, 2008a] contained a mistake regarding the previous statement. Namely, the way in which ll-dependency pairs were constructed could lead to narrowing chains which did not correspond to any narrowing derivation. In the definition of ll-dependency pair contained in this thesis, the echoing subterm is replaced by a $\bullet(x_1 \ldots x_n)$ term. In the original version from [*Alpuente et al.*, 2008a], this replacement was not performed. This Example 4.22 below demonstrates how this leads to false chains.

Example 4.22. *Consider the TRS defined by the rules:*

$$\begin{aligned} \mathtt{f}(\mathtt{g}(x)) &\to x \\ \mathtt{g}(x) &\to \mathtt{h}(x) \\ \mathtt{h}(x) &\to \mathtt{f}(\mathtt{g}(x)) \end{aligned}$$

With the original definition of narrowing dependency pair introduced in [Alpuente et al., 2008a], the set of pairs obtained for these rules is:

$$\begin{aligned} \mathtt{F}(\mathtt{g}(x)) &\to \mathtt{G}(x) && (ll) && (4.1) \\ \mathtt{G}(x) &\to \mathtt{H}(x) && (lr) && (4.2) \\ \mathtt{H}(x) &\to \mathtt{F}(\mathtt{g}(x)) && (lr) && (4.3) \\ \mathtt{H}(x) &\to \mathtt{G}(x) && (lr) && (4.4) \end{aligned}$$

The sequence (4.4),(4.2),(4.4),(4.2), ..., constitutes an infinite narrowing chain, as there is a $\leadsto_{\mathcal{R}\cup\mathcal{P}}$ derivation:

$$\mathtt{H}(x) \leadsto_{(4.4)} \mathtt{G}(x) \leadsto_{(4.2)} \mathtt{H}(x) \leadsto_{(4.4)} \cdots$$

This chain is mirrored[4] *by the following narrowing $\leadsto_{\mathcal{R}}$ derivation:*

$$\mathtt{h}(x) \leadsto_{\mathcal{R}} \mathtt{f}(\mathtt{g}(x)) \leadsto_{\mathcal{R}} \mathtt{f}(\mathtt{h}(x)) \leadsto_{\mathcal{R}} \mathtt{f}(\mathtt{f}(\mathtt{g}(x))) \leadsto_{\mathcal{R}} \cdots$$

[4]The precise meaning of "mirrored" in this sentence will become clear with Lemma 4.23 below.

The sequence (4.3), (4.1), (4.2), (4.3), (4.1),(4.2)..., also constitutes a narrowing chain, as witnessed by the following $\leadsto_{\mathcal{R}\cup\mathcal{P}}$ *derivation:*

$$\mathtt{H}(x) \leadsto_{(4.3)} \mathtt{F}(\mathtt{g}(x)) \leadsto_{(4.1)} \mathtt{G}(x) \leadsto_{4.2} \mathtt{H}(x) \leadsto_{4.3} \cdots$$

However, the reader can check that there is no infinite $\leadsto_{\mathcal{R}}$ *derivation which mirrors this chain, because there is no way to capture the echoing term* $\mathtt{g}(x)$ *in the step given with the first rule corresponding to the ll-dependency pair 4.1:*

$$\mathtt{c}(\mathtt{h}(x), x) \leadsto_{\mathcal{R}} \mathtt{c}(\mathtt{f}(\mathtt{g}(x)), x) \leadsto_{\mathcal{R}} \mathtt{c}(x, x) \not\leadsto_{\mathcal{R}}$$

Therefore the narrowing chains of [Alpuente et al., 2008a] overapproximate the narrowing derivations of the system, which is clearly undesirable. With the narrowing dependency pairs of Definition 4.17, the set of pairs obtained for these rules is the same except for the ll-dependency pair (4.1)*:*

$$\begin{array}{rcll} \mathtt{F}(\bullet(x)) & \to & \mathtt{G}(x) & (ll) \qquad (4.1') \\ \mathtt{G}(x) & \to & \mathtt{H}(x) & (lr) \qquad (4.2) \\ \mathtt{H}(x) & \to & \mathtt{F}(\mathtt{g}(x)) & (lr) \qquad (4.3) \\ \mathtt{H}(x) & \to & \mathtt{G}(x) & (lr) \qquad (4.4) \end{array}$$

Now the sequence (4.3), (4.1), (4.2), (4.3), (4.1),(4.2)... simply does not constitute a narrowing chain.

$$\mathtt{H}(x) \leadsto_{(4.3)} \mathtt{F}(\mathtt{g}(x)) \not\leadsto_{(4.1)}$$

The following result formalizes the soundness of analyzing narrowing chains.

Lemma 4.23. *Let* $\mathcal{R}$ *be a GTRS. Given a* ($NDP_{\mathcal{R}}$,$\mathcal{R}$)*–narrowing chain* $s_1^\# \to t_1^\#$, $s_2^\# \to t_2^\#, \ldots$, *there exists a narrowing derivation in* $\mathcal{R}$ *which gives at least one reduction step for every pair in the chain.*

Namely, there is a term t_0, *contexts* $C_1[\Box], C_2[\Box], \ldots$, *positions* $p_0, p_1, \ldots$, *rules* $l_1 \to r_1, l_2 \to r_2, \ldots$ *from* $\mathcal{R}$, *and substitutions* $\sigma_1, \sigma_2, \ldots$ *and* $\tau_2, \tau_3, \ldots$, *such that there is a derivation:*

$$t_0 \overset{p_0}{\leadsto}_{\sigma_1, l_1\to r_1} C_1[t_1\sigma_1]_{p_1} \overset{>p_1 *}{\leadsto}_{\tau_2,\mathcal{R}} \circ \overset{p_1}{\leadsto}_{\sigma_2, l_2\to r_2} C_2[t_2\sigma_2]_{p_2} \overset{>p_2 *}{\leadsto}_{\tau_3,\mathcal{R}} \circ \overset{p_2}{\leadsto}_{\sigma_3, l_3\to r_3} \cdots$$

where the rule $l_i \to r_i \in \mathcal{R}$ *is:*

- *If* $s_i^\# \to t_i^\#$ *is a lr-dependency pair, then* $l_i = s_i$ *and* $r_i|_p = t_i$,
- *If* $s_i^\# \to t_i^\#$ *is an ll-dependency pair with* $s_i^\#|_p = \bullet(Var(s_i|_p))$, *then* $l_i \to r_i$ *is a rule*[5] *such that* $l_i = s_i[t_i]_p$.

Proof. By induction on the length n of the chain. The case $n = 0$ is immediate. For $n > 0$, let us assume that the statement holds for the initial part of the chain, i.e. there is a narrowing derivation for the chain $s_1^\# \to t_1^\#, \ldots, s_{n-1}^\# \to t_{n-1}^\#$, and then

[5]Note that if there is more than one such rule, the choice is irrelevant.

prove the statement for the whole chain by extending this derivation with the segment corresponding to the pair $s_n^{\#} \to t_n^{\#}$.

The induction hypothesis yields a term t_0, contexts $C_1[\ ], \ldots, C_{n-1}[\ ]$, positions $p_1, \ldots, p_{n-1}$, and substitutions $\sigma_1, \ldots, \sigma_{n-1}$ s.t. there is a $(\leadsto_{\mathcal{R}})$-derivation:

$$t_0 \overset{p_0}{\leadsto}_{\sigma_1,\mathcal{R}} C_1[t_1\sigma_1]_{p_1} \overset{>p_1 *}{\leadsto}_{\tau_2,\mathcal{R}} \circ \overset{p_1}{\leadsto}_{\sigma_2,\mathcal{R}} \cdots \overset{>p_{n-2} *}{\leadsto}_{\tau_{n-1},\mathcal{R}} \circ \overset{p_{n-2}}{\leadsto}_{\sigma_{n-1},\mathcal{R}} C_{n-1}[t_{n-1}\sigma_{n-1}]_{p_{n-1}}$$

And the segment to be shown is the following:

$$C_{n-1}[t_{n-1}\sigma_{n-1}]_{p_{n-1}} \overset{>p_{n-1} *}{\leadsto}_{\tau_n,\mathcal{R}} \circ \overset{p_{n-1}}{\leadsto}_{\sigma_n, l_n \to r_n} C_n[t_n\sigma_n]_{p_n}$$

The definition of narrowing chain yields the following assumption: there exist substitutions τ' and σ' such that there is a $(\leadsto_{NDP_{\mathcal{R}} \cup \mathcal{R}})$-derivation:

$$t_{n-1}^{\#}\sigma_{n-1} \overset{>\epsilon *}{\leadsto}_{\tau',\mathcal{R}} u^{\#} \overset{\epsilon}{\leadsto}_{\sigma', s_n^{\#} \to t_n^{\#}} t_n^{\#}\sigma' \tag{4.5}$$

We consider the two kinds of narrowing dependency pair separately.

- $s_n^{\#} \to t_n^{\#}$ is an lr-dependency pair.

  Then there is a rule $s_n \to r_n \in \mathcal{R}$ and a position q s.t. $r_n|_q = t_n$. Defining $\tau_n = \tau'$ and $\sigma_n = \sigma'$, by (4.5) there is a derivation:

  $$C_{n-1}[t_{n-1}\sigma_{n-1}]_{p_{n-1}} \overset{>p_{n-1} *}{\leadsto}_{\tau_n,\mathcal{R}} C_{n-1}\tau_n[u]_{p_{n-1}} \overset{p_{n-1}}{\leadsto}_{\sigma_n, s_n \to r_n} C_{n-1}\tau_n\sigma_n[r_n\sigma_n]_{p_{n-1}} = \beta$$

  Now we just have to define $p_n = p_{n-1}.q$ and take $C_n = \beta[\Box]_{p_n}$ to complete the proof for this case.

- $s_n^{\#} \to t_n^{\#}$ is an ll-dependency pair with $s_n^{\#}|_p = \bullet(x_1 \ldots x_n)$.

  Then there is at least one rule $l_n \to r_n \in \mathcal{R}$ such that $l_n = s_n[t_n]_p$.

  As $\bullet$ does not occur in $\mathcal{R}$ nor in the right-hand sides of the rules of $NDP_{\mathcal{R}}$, it follows from (4.5) that the subterm $u^{\#}|_p$ must necessarily be a variable x, and that $x\sigma' = \bullet(x_1 \ldots x_n)$. Since we want to give a step with the rule $l_n \to r_n$ instead of the pair $s_n^{\#} \to t_n^{\#}$, define $\sigma_n(x) = t_n\sigma_n$ and $\sigma_n(x') = \sigma'(x')$ for any variable x' distinct from x. It follows that $u\sigma_n = l\sigma_n$, and by letting $\tau_n = \tau'$, (4.5) yields the derivation

  $$C_{n-1}[t_{n-1}\sigma_{n-1}]_{p_{n-1}} \overset{>p_{n-1} *}{\leadsto}_{\tau_n,\mathcal{R}} C_{n-1}\tau_n[u]_{p_{n-1}} \overset{p_{n-1}}{\leadsto}_{\sigma_n, l_n \to r_n} C_{n-1}\tau_n\sigma_n[r_n\sigma_n]_{p_n} = \beta$$

  Now it only remains to be shown that $\beta = C_n[t_n\sigma_n]_{p_n}$. For this, choose t_0 so that $x \in Var(C_{n-1}\tau_n)$. Observe that this is always possible, as we have freedom to choose t_0. For instance, given a t_0' for which this is not satisfied, define $t_0 = \mathsf{c}(t_0', y)$ for a fresh binary symbol c and a variable $y \in Var(t_0')$ s.t. $y\sigma_1\tau_2\sigma_2\tau_3 \ldots \tau_{n-1}\sigma_{n-1} \trianglerighteq x$. Given an occurrence of x in $C_{n-1}\tau_n$, $q \in \mathcal{P}os_x(C_{n-1}\tau_n)$, define $p_n = q$ and $C_n = \beta[\Box]_{p_n}$. This concludes the proof. $\Box$

It can be seen from Lemma 4.23 that each narrowing step given with the rule corresponding to an lr-dependency pair is a TOP step, whereas each narrowing step given with the rule corresponding to an ll–dependency pair is either a HYBRID or ECHOING step. Equivalence between the first and third statements now follows by transitivity.

The following proposition establishes a lifting between narrowing and rewriting chains. It will be highly useful in Section 4.3, where we adapt several DP processors from the rewriting setting to the narrowing setting. The proposition simply states that every rewriting chain is also a narrowing chain, be it finite or infinite; but the other way around is only true when we consider a finite narrowing chain, or a finite prefix of an infinite narrowing chain. Intuitively, this is because the substitution σ computed by an infinite narrowing derivation can contain infinite terms in the limit, whereas rewriting chains are assumed to contain only finite terms (cf. Definition 4.3).

Proposition 4.24. *Given a TRS $\mathcal{R}$ with no extra variables, every rewriting chain $(\mathcal{P}, \mathcal{R})$ is also a narrowing chain. Moreover, if the rewriting chain is infinite then the associated narrowing chain is also infinite.*

Conversely, given a GTRS $\mathcal{R}$ (possibly with extra variables), every finite prefix of a narrowing chain $(\mathcal{P}, \mathcal{R})$ is also a rewriting chain[6]*, and minimality is preserved.*

Proof. First we show first that every rewriting chain has an associated narrowing chain.

Let $s_0 \to t_0, s_1 \to t_1, \ldots$ be a possibly infinite, minimal rewriting chain. By definition, there exists a substitution σ s.t. $t_i\sigma \to^*_{\mathcal{R}} s_{i+1}\sigma$, for every two consecutive pairs of the chain $s_i \to t_i, s_{i+1} \to t_{i+1}$. Then we can construct a $\to_{\mathcal{R}\cup\mathcal{P}}$ derivation of the form:

$$s_0\sigma \overset{\epsilon}{\to}_{s_0\to t_0} t_0\sigma \overset{>\epsilon *}{\to}_{\mathcal{R}} s_1\sigma \overset{\epsilon}{\to}_{s_1\to t_1} t_1\sigma \overset{>\epsilon *}{\to}_{\mathcal{R}} s_2\sigma \overset{\epsilon}{\to}_{s_2\to t_2} \cdots$$

We can apply the lifting lemma of Proposition 2.1 (with $\theta = id$) to extract a narrowing derivation of the form:

$$s_0\sigma \overset{\varepsilon}{\leadsto}_{\sigma,s_0\to t_0} t_0\sigma \overset{>\epsilon *}{\leadsto}_{id,\mathcal{R}} s_1\sigma \overset{\varepsilon}{\leadsto}_{\sigma,s_1\to t_1} t_1\sigma \overset{>\epsilon *}{\leadsto}_{id,\mathcal{R}} \cdots$$

which constitutes a narrowing chain.

Now we show the converse, that every finite prefix of a narrowing chain has an associated rewriting chain. We proceed in the same way. Let $\mathcal{P} = s_0 \to t_0, s_1 \to t_1, \ldots$. Then there is a substitution σ_0 such that the following $\leadsto_{\mathcal{P}\cup\mathcal{R}}$ derivation exists:

$$s_0\sigma_0 \overset{\epsilon}{\leadsto}_{\sigma_0,s_0\to t_0} t_0\sigma_0 \overset{>\epsilon *}{\leadsto}_{\rho_0,\mathcal{R}} s_1\sigma_1 \overset{\epsilon}{\leadsto}_{\sigma_1,s_1\to t_1} t_1\sigma_1 \overset{>\epsilon *}{\leadsto}_{\rho_1,\mathcal{R}} \cdots$$

Consider a *finite* prefix of the derivation above. In order to prove that this prefix is a rewriting chain, we must show that there exists a substitution σ containing only finite terms such that $t_i\sigma \to^*_{\mathcal{R}} s_{i+1}\sigma$ for every two consecutive pairs in the chain. By the soundness of narrowing (cf. Proposition 2.2) this substitution exists,

[6]Note that rewriting chains were introduced for TRSs only, while in general $\mathcal{R}$ and $\mathcal{P}$ are generalized TRSs (i.e. with extra variables). But this is no problem since the definition of rewriting chain can be extended to generalized TRSs without changes.

$\sigma = \sigma_0\rho_0\sigma_1\rho_1\ldots$. Moreover if this is a finite chain, or a finite prefix of an infinite chain, then σ has a finite domain and contains only finite terms. That minimality is preserved follows from the fact that $\rightsquigarrow$-termination implies $\rightarrow$-termination (cf. Proposition 2.3). □

Now we are able to show that, whenever there are no infinite narrowing chains, narrowing does terminate.

Theorem 4.25 (Termination Criterion)**.** *A GTRS $\mathcal{R}$ is terminating for narrowing if and only if no infinite minimal ($NDP_{\mathcal{R}}$,$\mathcal{R}$)–narrowing chain exists.*

Proof. The *if* case is straightforward from Lemma 4.15 and the *only if* case is straightforward from Lemma 4.23. □

The following definition lifts the standard notion of dependency graph (cf. Definition 4.5) to the narrowing setting by considering narrowing dependency pairs and chains.

Definition 4.26 (Narrowing Dependency Graph)**.** *Given a set of rules (possibly with extra variables $\mathcal{R}$ and a set of pairs (possibly with extra variables) $\mathcal{P}$, the narrowing dependency graph is the directed graph where the nodes are the elements of $\mathcal{P}$, and there is an edge from $s \to t \in \mathcal{P}$ to $u \to v \in \mathcal{P}$ if $s \to t$, $u \to v$ is a narrowing chain from $\mathcal{P}$.*

The lemma below establishes that the narrowing dependency graph and the dependency graph coincide in absence of extra variables (otherwise, the rewriting dependency graph is not defined), as an immediate consequences from the lifting between narrowing and rewriting chains.

Lemma 4.27. *Given a set of rules $\mathcal{R}$ satisfying the variable condition and a set of pairs $\mathcal{P}$, the narrowing dependency graph is the same as the dependency graph.*

Proof. Follows from the lifting between narrowing chains and rewriting chains stated in Proposition 4.24. □

For finite GTRSs, infinite chains show up as cycles in the dependency graph[7]. Unfortunately, it is well known that computing the exact dependency graph is an undecidable problem and thus several approximations [*Arts and Giesl*, 2000; *Giesl et al.*, 2006b; *Hirokawa and Middeldorp*, 2005; *Thiemann*, 2007] are used to compute an *estimated* dependency graph which includes the exact graph. The following approximation is commonly used.

Definition 4.28 (Estimated Dependency Graph)**.** [*Arts and Giesl*, 2000] *Let $\mathcal{R}$ be a set of rules (possibly with extra variables), and $\mathcal{P}$ be a set of pairs (possibly with extra variables). Let* $\textsc{cap}_{\mathcal{R}}(t)$ *be the result of replacing*[8] *all the proper subterms of t with a*

[7]The converse does not hold, not every cycle corresponds to an infinite chain.
[8]This function was first defined for approximating loops in dependency graphs in [*Alpuente et al.*, 1994], where it is called $\overset{\circ}{t}$.

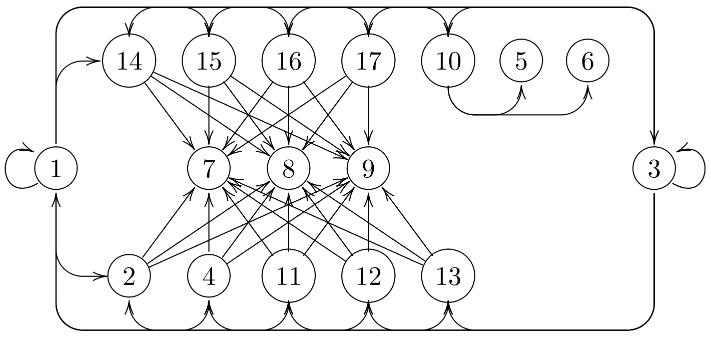


Figure 4.3: Estimated dependency graph of $\mathcal{R}_{Policy}$

defined root symbol by a fresh variable, and $\textsc{ren}(t)$ *the linearization of* t *(replacing all ocurrences of a non linear variable with independent fresh variables). The nodes of the estimated dependency graph (EDG) are the pairs of* $\mathcal{P}$ *and there is an edge from* $s^\# \to t^\#$ *to* $u^\# \to v^\#$ *iff* $\textsc{ren}(\textsc{cap}_\mathcal{R}(t))$ *and* u *are unifiable.*

As in the rewriting setting, this estimation correctly overapproximates the exact narrowing dependency graph.

Lemma 4.29. *Let* $\mathcal{R}$ *be a set of rules (possibly with extra variables) and* $\mathcal{P}$ *be a set of pairs (possibly with extra variables). Define EDG to be the estimation of its dependency graph G, obtained as in definition 4.28 above. Then EDG contains G, i.e. every node in G is a node in EDG and every edge in G is also an edge in EDG.*

Proof. By straighforward generalization of the proof of Theorem 21 in [Arts and Giesl, 2000] to the case of extra variables in $\mathcal{P}$ and $\mathcal{R}$, in the sense that no changes at all are required. $\square$

Example 4.30. *For the TRS* $\mathcal{R}_{Policy}$ *of Example 4.1 and the set of DPs obtained in Example 4.19, the EDG is shown in Figure 4.3.*

4.2.3 Classes of TRSs where narrowing enjoys the TRAT property

We take a short detour to outline that, as a byproduct of the results in this section, we are now able to give new characterizations of classes of TRSs where narrowing enjoys the TRAT property.

The proposition below highlights the role that ll-dependency pairs play in non TRAT systems. Essentially, it states that saying that narrowing enjots the TRAT property in a system $\mathcal{R}$ is equivalent to saying that $\mathcal{R}$ gives rise to no ll-dependency pairs, or at least they play no role in minimal infinite derivations.

Proposition 4.31. *Let* $\mathcal{R}$ *be a TRS.The following three statements are equivalent.*

1. *$\rightsquigarrow_{\mathcal{R}}$ has the* TRAT *property for any term,*

2. *every infinite minimal $\rightsquigarrow_{\mathcal{R}}$ derivation is composed solely of* TOP *steps, and*

3. *there are no ll-dependency pairs ocurring in any minimal infinite $\rightsquigarrow_{\mathcal{R}}$ chain.*

Proof. That the first and second statements are equivalent follows from the classification of infinite narrowing derivations in Lemma 4.15. That the second and third statements are equivalent follows from Lemma 4.23, since non TOP steps, i.e., HYBRID or ECHOING steps, correspond always to an ll–dependency pair. □

While the proposition above completely characterizes the class of systems that have the TRAT property for narrowing, it is not useful for deciding whether a given TRS enjoys the TRAT property. An approximation based on the EDG can be used instead.

Corollary 4.32. *Let $\mathcal{R}$ be a TRS. Narrowing has the* TRAT *property in $\mathcal{R}$ if no ll-dependency pair appears in a cycle of (an over-estimation of) the dependency graph of $\mathcal{R}$.*

Proof. Follows from Proposition 4.31. □

It also follows that narrowing enjoys the TRAT property in the rnf-based systems studied in Chapter 3, which constitute a superclass of constructor systems. This generalizes the results in [*Nishida et al.*, 2003] for constructor systems.

Corollary 4.33. *Let $\mathcal{R}$ be a TRS. If $\mathcal{R}$ is* srnf*-based, then the narrowing relation $\rightsquigarrow_{\mathcal{R}}$ has the* TRAT *property.*

Proof. Let $\mathcal{R}$ be a srnf-based TRS, and $l^{\#}[\bullet(x_1 \ldots x_n)]_p \to (l|_p)^{\#}$ a ll-dependency pair extracted from some rule $l \to r \in \mathcal{R}$. By definition, $l|_p$ is a stable rigid normal form (cf. Definition 3.29), i.e. there is no substitution σ such that $l|_p\sigma \rightsquigarrow_{\mathcal{R}} t$, for any term t.

Hence ll-dependency pairs can only appear as the last element of a narrowing chain, and therefore never in an infinite chain. The result now follows from Proposition 4.31. □

It is also easy to show that narrowing enjoys the TRAT property in systems which satisfy the (*quasi stable rigidly normalized condition*) of Chapter 3 (cf. Definition 3.42). This class includes, for instance, the rnf-based systems of Section 3.2.1 and the left-plain systems of Section 3.4.

Proposition 4.34. *Let $\mathcal{R}$ be a TRS that satisfies the QSRN condition. Then $\rightsquigarrow_{\mathcal{R}}$ enjoys the* TRAT *property.*

Proof. Follows from Lemma 4.15. Let $\mathcal{R}$ be a TRS that satisfies the QSRNC. By definition, the substitution σ computed in a $\rightsquigarrow_{\mathcal{R}}$ step cannot introduce non-constructor subterms from the left hand side of a rule. From this observation we can conclude, by Corollary 4.16, that no substitution computed in a $\rightsquigarrow_{\mathcal{R}}$ step can introduce a binding

$u \in \mathcal{T}^{\circlearrowleft}$ or $u \in \mathcal{T}^{\infty}_{\rightsquigarrow}$, which precludes HYBRID and ECHOING steps, respectively. Hence every infinite $\rightsquigarrow_{\mathcal{R}}$ derivation is composed solely of TOP steps, and therefore $\mathcal{R}$ does enjoy the TRAT property. $\square$

4.2.4 Proving the Absence of Infinite Narrowing Chains

In this section we consider the problem of proving that a narrowing chain is finite. The approach followed is inspired by [Nishida et al., 2003] but we provide all results without requiring TRAT. Essentially, it consists on finding a mapping from narrowing chains to rewriting chains, such that for each narrowing chain, finiteness of its associated rewriting chain implies finiteness of the narrowing chain. Then, we use standard techniques from the rewriting setting to prove that the rewriting chain is finite.

Let us start by recalling the notion of argument filtering. Intuitively speaking, argument filterings map every function symbol to a subset of its argument positions, i.e., given a function symbol $\mathtt{f}$ of arity n, $\pi(\mathtt{f})$ returns a subset of $\{1,\ldots,n\}$ so that the arguments whose position is not in the set are filtered away. This notion can also be used as a function to apply this mapping to a term t, with the homomorphic lifting to rules and GTRSs.

Definition 4.35 (Argument Filtering)**.** [Arts and Giesl, 2000] *An argument filtering (AF) for a signature Σ is a mapping π that assigns to every n-ary function symbol $f \in \Sigma$ an argument position $i \in \{1,\ldots,n\}$, or a (possibly empty) list $[i_1,\ldots,i_m]$ of argument positions with $1 \le i_i < \ldots < i_m \le n$. The signature Σ_π consists of all function symbols $\mathtt{f}$ s.t. $\pi(\mathtt{f})$ is some list $[i_1,\ldots,i_m]$, where in Σ_π the arity of $\mathtt{f}$ is m. Every AF π induces a mapping from $\mathcal{T}(\Sigma,\mathcal{V})$ to $\mathcal{T}(\Sigma_\pi,\mathcal{V})$:*

$$\pi(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \pi(t_i) & \text{if } t = \mathtt{f}(t_1,\ldots,t_n) \text{ and } \pi(\mathtt{f}) = i \\ \mathtt{f}(\pi(t_{i_1}),\ldots,\pi(t_{i_m})) & \text{if } t = \mathtt{f}(t_1,\ldots,t_n) \text{ and } \pi(\mathtt{f}) = [i_1,\ldots,i_m] \end{cases}$$

- *By abuse of notation, we often simply write Σ instead of Σ_π, and keep the same symbol for the original function and the filtered function with a possibly different arity.*
- *An argument filtering is called* collapsing *if $\pi(f) = i$ for some f.*
- *Argument filterings can be homomorphically extended to TRSs and substitutions:*

$$\pi(\mathcal{R}) = \{\pi(l) \to \pi(r) \mid l \to r \in \mathcal{R}\} \qquad \pi(\sigma)(x) = \pi(\sigma(x))$$

- *For any filtering π and relation $\succ$, we let $\succ_\pi$ be the relation where $t \succ_\pi u$ holds iff $\pi(t) \succ \pi(u)$.*

Example 4.36. *Consider the TRS of Example 4.1 and the term*

$$t = \mathtt{filter}(\mathtt{pckt}(\mathtt{secure}, dst, \mathtt{new}))$$

Given the argument filtering $\pi_1(\texttt{pckt}) = [1,3]$ *and* $\pi_1(f) = [1,\ldots,ar(f)]$ *for any other* $f \in \Sigma$*, we have that* $\pi_1(t) = \texttt{filter}(\texttt{pckt}(\texttt{secure},\texttt{new}))$*.*

Given the argument filtering $\pi_2(\texttt{pckt}) = 2$*,* $\pi_2(\texttt{filter}) = 1$*, and* $\pi_2(f) = [1,\ldots, ar(f)]$ *for any other* $f \in \Sigma$*, we have that* $\pi_2(t) = dst$*.*

Our main result in this section relates infinite narrowing $(\mathcal{P},\mathcal{R})$–chains to infinite rewriting $(\pi(\mathcal{P}),\pi(\mathcal{R}))$–chains. In order to prove this result, we need to prove an auxiliary lemma first. The lemma establishes a correspondence between rewriting derivations in $\mathcal{R}$ and derivations in the filtered TRS $\pi(\mathcal{R})$. For this purpose we make use of the auxiliary notion of *regarded positions* w.r.t. an argument filtering found in the literature, e.g. in [Thiemann, 2007]. In essence, the regarded positions of t w.r.t. π are those positions of t which are not dropped by the filtering π.

Definition 4.37 (Regarded Positions [Thiemann, 2007])**.** *Given an argument filtering* π *and a term* t*, we let* $RegPos_\pi(t)$ *denote the* regarded positions *of* t *w.r.t.* π*, defined recursively as:*

$$RegPos_\pi(t) = \{\epsilon\} \cup \{i.p \mid t = \mathtt{f}(t_1,\cdots,t_n), p \in RegPos_\pi(t_i), i \in \pi(f)\}$$

Lemma 4.38 (Properties of argument filterings)**.** *Let* $\mathcal{R}$ *be a GTRS over a signature* Σ*,* π *be an argument filtering over* Σ*,* $t,s \in \mathcal{T}(\Sigma,\mathcal{V})$ *be terms, and* σ *be a substitution.*

(a) $\pi(t\sigma) = \pi(t)\pi(\sigma)$*.*

(b) If $s \xrightarrow{p}_{l\to r} t$ *and* $p \in RegPos_\pi(t)$ *then* $\pi(s) \xrightarrow{\pi}_{\pi(l)\to\pi(r)} (t)$*, otherwise* $\pi(s) = \pi(t)$*.*

(c) If $s \to^*_{\mathcal{R}} t$ *then* $\pi(s) \to^*_{\pi(\mathcal{R})} \pi(t)$*. Moreover, we can assume that the derivation in* $\pi(\mathcal{R})$ *uses the same rules in the same order at the corresponding filtered positions (whenever the filtered position exists).*

Proof. (a) We perform induction on t. If t is a variable x then $\pi(t\sigma) = \pi(\sigma(x)) = t\pi(\sigma) = \pi(t)\pi(\sigma)$.

If $t = \mathtt{f}(t_1,\ldots,t_n)$ there are two cases. If $\pi(\mathtt{f}) = [i_1,\ldots,i_m]$, then $\pi(t\sigma) = \pi(\mathtt{f}(t_1\sigma,\ldots,t_n\sigma))$, by definition of argument filtering $\pi(t\sigma) = \mathtt{f}(\pi(t_{i_1}\sigma),\ldots, \pi(t_{i_m}\sigma))$, by the induction hypothesis $\pi(t\sigma) = \mathtt{f}(\pi(t_{i_1})\pi(\sigma)),\ldots,\pi(t_{i_m})\pi(\sigma))$, and by reversing substitution application $\pi(t\sigma) = \mathtt{f}(\pi(t_{i_1}),\ldots,\pi(t_{i_m}))\pi(\sigma) = \pi(t)\pi(\sigma)$.

Otherwise $\pi(\mathtt{f}) = i$, $\pi(t\sigma) = \pi(t_i\sigma)$. By the induction hypothesis, $\pi(t_i\sigma) = \pi(t_i)\pi(\sigma) = \pi(t)\pi(\sigma)$, and we are done.

(b) We perform induction on the position of the reduction. We assume that s is of the form $\mathtt{f}(s_1,\ldots,s_n)$. If $p = \epsilon$ then $p \in RegPos_\pi(s)$ by definition. We have $s = l\sigma \to_{l\to r} r\sigma = t$ and $\pi(s) = \pi(l\sigma)$. By (a), $\pi(s) = \pi(l)\pi(\sigma)$. Now it is immediate that $\pi(l)\pi(\sigma) \to_{\pi(l)\to\pi(r)} \pi(r)\pi(\sigma) = \pi(r\sigma) = \pi(t)$, where again we make use of (a) in the last step.

Otherwise $p = i.j$, $s = \mathtt{f}(s_1\ldots s_i\ldots s_n) \to_{l\to r} \mathtt{f}(s_1\ldots v_i\ldots s_n) = t$, and $s_i = l\sigma, v_i = r\sigma$.

If $i \in \pi(\mathtt{f})$ then $p \in \mathit{RegPos}_\pi(s)$, so the induction hypothesis implies $\pi(s_i) \to_{\pi(l)\to\pi(r)} \pi(v_i)$. As $\to_\mathcal{R}$ is *compatible* with the Σ operations, the following fact holds:

$$\mathtt{f}(\pi(s_1)\dots\pi(s_i)\dots\pi(s_n)) \to_{\pi(l)\to\pi(r)} \mathtt{f}(\pi(s_1)\dots\pi(v_i)\dots\pi(s_n))$$

and therefore we obtain $\pi(s) \to_{\pi(l)\to\pi(r)} \pi(t)$.

If $i \notin \pi(\mathtt{f})$ and $p \notin \mathit{RegPos}_\pi(s)$, we have that $\pi(s) = \pi(t)$, which concludes the proof of claim b.

(c) Follows from b by performing induction on the length of the derivation. □

We are now in position to introduce the result relating infinite narrowing and rewriting chains, allowing us to prove the absence of narrowing chains by analyzing standard rewriting chains. This is obviously very useful, as it means that one can reuse all the DP termination techniques available in the literature of rewriting to prove the termination of narrowing.

Theorem 4.39. *Let $\mathcal{R}$ be a GTRS over a signature Σ, $\mathcal{P}$ be a GTRS over a signature $\Sigma^\#$, and π an argument filtering over $\Sigma^\#$ s.t. $\pi(\mathcal{R})$ and $\pi(\mathcal{P})$ are TRSs, and there is at least one pair $s \to t \in \mathcal{P}$ such that $\pi(t)$ is ground.*

If $(\mathcal{P},\mathcal{R})$ has an infinite narrowing chain, and $s \to t$ occurs at least once in the chain, then $(\pi(\mathcal{P}),\pi(\mathcal{R}))$ has an infinite rewriting chain.

Proof. We prove the claim by means of a simulation between the narrowing derivation and the filtered rewriting derivation.

We assume that $\pi(t_1)$ is ground for the first pair $s_1 \to t_1 \in \mathcal{P}$. There is no loss of generality as we can always skip a prefix of $\mathcal{P}$, and still have an infinite $(\mathcal{P}$,$\mathcal{R})$–narrowing chain.

By definition of narrowing chain, there is an infinite $\rightsquigarrow_{\mathcal{P}\cup\mathcal{R}}$ derivation of the form:

$$u_1 \overset{\epsilon}{\rightsquigarrow}_{\sigma_1, s_1\to t_1} u_2 \overset{>\epsilon *}{\rightsquigarrow}_{\tau_1,\mathcal{R}} u_3 \overset{\epsilon}{\rightsquigarrow}_{\sigma_2, s_2\to t_2} u_4 \overset{>\epsilon *}{\rightsquigarrow}_{\tau_2,\mathcal{R}} \cdots$$

Then by the soundness of narrowing (cf. Proposition 2.2) there is also a $\to_{\mathcal{R}\cup\mathcal{P}}$ derivation of the form:

$$u_1\theta \overset{\epsilon}{\to}_{s_1\to t_1} u_2\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} u_3\theta \overset{\epsilon}{\to}_{s_2\to t_2} u_4\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} \cdots$$

where θ is a substitution possibly carrying infinite terms.

By (c) in Lemma 4.38, there is also a $\to_{\pi(\mathcal{R}\cup\mathcal{P})}$ derivation of the form:

$$\pi(u_1\theta) \overset{\epsilon}{\to}_{\pi(s_1)\to\pi(t_1)} \pi(u_2\theta) \overset{>\epsilon *}{\to}_{\pi(\mathcal{R})} \pi(u_3\theta) \overset{\epsilon}{\to}_{\pi(s_2)\to t_2} \pi(u_4\theta) \overset{>\epsilon *}{\to}_{\pi(\mathcal{R})} \cdots$$

Since $\pi(t_1)$ is ground by assumption, we can apply (a) from Lemma 4.38 to obtain $\pi(u_2\theta) = \pi(t_1)\sigma_1\theta = \pi(t_1)$, and since $\pi(\mathcal{R})$ and $\pi(\mathcal{P})$ have no extra variables, we can simplify the filtered derivation as follows:

$$\pi(u_1\theta) \overset{\epsilon}{\to}_{\pi(s_1)\to\pi(t_1)} \pi(t_1) \overset{>\epsilon *}{\to}_{\pi(\mathcal{R})} \pi(u_3) \overset{\epsilon}{\to}_{\pi(s_2)\to\pi(t_2)} \pi(u_4) \overset{>\epsilon *}{\to}_{\pi(\mathcal{R})} \cdots$$

From this derivation it follows that $\pi(s_1) \to \pi(t_1), \pi(s_2) \to \pi(t_2), \ldots$ constitutes a $(\pi(\mathcal{P}), \pi(\mathcal{R}))$-rewriting chain (the steps given with $\pi(\mathcal{P})$ are preserved by (b) in Lemma 4.38 since the root position is always a regarded position). As $\mathcal{P}$ is infinite, there are an infinite number of $\pi(\mathcal{P})$ steps in this filtered derivation, and we are done. □

4.3 The Narrowing Dependency Pair framework

The DP framework (c.f. Section 4.1) is a formalism for simplifying the automation of DP based techniques in a modular way. We consider in this section its generalization to the narrowing relation, and introduce a number of NDP processors based on the theoretical results developed in the previous sections of this chapter.

Definition 4.40 (NDP problems and processors)**.** *An NDP problem is a tuple* $\langle \mathcal{P}, \mathcal{R}, f\rangle_{\leadsto}$ *consisting of two GTRSs* $\mathcal{R}$ *and* $\mathcal{P}$ *and a minimality flag* $f \in \{\mathbf{m}, \mathbf{a}\}$*, where* **m** *and* **a** *stand for minimal and arbitrary respectively. We say that an NDP problem* $\langle \mathcal{P}, \mathcal{R}, f\rangle_{\leadsto}$ *is finite if there is no associated infinite (minimal if* f *is* **m***)* $(\mathcal{P}, \mathcal{R})$*-narrowing chain. An NDP problem* $\langle \mathcal{P}, \mathcal{R}, f\rangle_{\leadsto}$ *is infinite if it is not finite, or if* $\leadsto_{\mathcal{R}}$ *is not terminating.*

An NDP processor *is a function Proc which takes an NDP problem and either returns a new set of (NDP/DP) problems or fails. Proc is sound if for any NDP problem* $\mathcal{M}$*,* $\mathcal{M}$ *is finite whenever all the problems in Proc*$(\mathcal{M})$ *are finite.*

As usual, we construct a tree whose root is labeled with the problem $\langle NDP_{\mathcal{R}}, \mathcal{R}, \mathbf{m}\rangle_{\leadsto}$ and whose nodes are produced by application of sound NDP processors. $\leadsto_{\mathcal{R}}$ is terminating if all the leaf nodes of this tree are finite.

In the usual style [*Vidal*, 2008; *Schneider-Kamp et al.*, 2009b], this section shows how to adapt the essential DP processors to the NDP framework, and then a processor that transforms an NDP problem into a DP problem is introduced.

While all the theoretical basis has been laid already in the previous sections of this chapter and this section is mostly routine, there are still significant challenges that need to be overcome before the approach can be made efficiently implementable. In particular, finding a way to deal with strongly connected components of pairs instead of cycles is a non trivial problem. Moreover, in this section we consider the notion of usable rules and extend the simulation between narrowing chains and filtered rewriting chains introduced in Theorem 4.39 to work also with usable rules.

The results in this section improve over the ones that originally appeared in [*Alpuente et al.*, 2008a]. As already mentioned, the NDP framework now is able to operate[9] on strongly connected components (SCCs) instead of cycles which makes it much more efficient. In addition, the notion of usable rules for narrowing chains and the extension of the simulation in Theorem 4.39 are also new contributions.

[9]based on suggestions contributed by [*Schneider-Kamp*, 2009]

$$\begin{tikzcd}
1 \arrow[loop left] \arrow[r, bend left] & 3 \arrow[l, bend left] \arrow[loop right]
\end{tikzcd}$$

Figure 4.4: The single SCC in the dependency graph of $\mathcal{R}_{Policy}$

4.3.1 The Dependency Graph processor

For finite GTRSs, infinite chains show up as cycles in the dependency graph[10]. We can analyze every chain separately, that is, every cycle in the dependency graph. Actually, we can focus on the strongly connected components of the graph. This is accomplished by the Dependency Graph processor.

Theorem 4.41 (Dependency Graph Processor). *Let $\tau = \langle \mathcal{P}, \mathcal{R}, f\rangle_{\leadsto}$ be an NDP problem with $\mathcal{P}$ a finite set of pairs, and let $\texttt{Proc}$ be the processor such that*

$$\texttt{Proc}(\tau) = \{\langle \mathcal{P}_1, \mathcal{R}, f\rangle_{\leadsto}, \ldots, \langle \mathcal{P}_n, \mathcal{R}, f\rangle_{\leadsto}\}$$

where $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are the sets of nodes of every strongly connected component in the estimated dependency graph. Then, $\texttt{Proc}$ is sound.

Proof. We prove the theorem by contradiction. Assume that $\texttt{Proc}$ is not sound. Then there is an NDP problem $\tau = \langle \mathcal{P}, \mathcal{R}, f\rangle_{\leadsto}$ such that every subproblem $\tau \in \texttt{Proc}(\tau)$ is finite whereas τ is not. Then there is an infinite chain $\mathcal{P}_1 \subseteq \mathcal{P}$ such that –since $\mathcal{P}$ is finite– a tail $B \subseteq \mathcal{P}_1$ of pairs is repeated infinitely. The nodes corresponding to B form a cycle in the dependency graph, and by Theorem 4.29 in the estimated dependency graph. This cycle must belong to some SCC; the subproblem $\tau_i \in \texttt{Proc}(\tau)$ corresponding to this SCC is not finite, since B is also an infinite chain of τ_i. Thus we reach a contradiction with the assumption that all the subproblems are finite, which proves the soundness of $\texttt{Proc}$. □

Example 4.42. *In the graph obtained in the EDG of Example 4.30, the only SCC consists of nodes (1) and (3). Thus the dependency graph processor deletes all the other dependency pairs, and returns the problem $\{\langle\{(1),(3)\}, \mathcal{R}, \boldsymbol{m}\rangle_{\leadsto}\}$, corresponding to the graph in Figure 4.4.*

By making use of the dependency graph processor, we are now able to prove the termination of Example 4.10. It has only one left dependency pair $\texttt{F}(\bullet(x)) \to \texttt{G}(x)$, giving the NDP problem:

$$\left\langle \texttt{F}(\bullet(x)) \;\to\; \texttt{G}(x), \quad \begin{array}{rcl} \texttt{f}(\texttt{g}(x)) & \to & x \\ \texttt{g}(x) & \to & x \end{array}, \quad \mathbf{m} \right\rangle_{\leadsto}$$

Its associated dependency graph is empty. Since there are no cycles in the empty graph, we can use the dependency graph processor of Theorem 4.41 to transform it into the empty problem, succesfully proving termination of narrowing.

[10]The converse does not hold, i.e. not every cycle corresponds to an infinite chain.

4.3.2 The Reduction Pair processor

We introduce now a processor that, given a suitable reduction pair $(\succeq, \succ)$ and an argument filtering π, decomposes a problem into two smaller problems, containing the same rules but less pairs than the original one. The reduction pair processor presented below is more efficient than the one originally presented in [*Alpuente et al.*, 2008a], as it is able to operate on SCCs of pairs instead of being restricted to cycles. This is highly desirable for efficiency reasons [*Hirokawa and Middeldorp*, 2005], namely since in the worst case there are $2^n - 1$ cycles for n pairs, while there are at most n SCCs.

In the following, for any GTRS $\mathcal{P}$ and relation $\succ$, let $\mathcal{P}_\succ = \{s \to t \in \mathcal{P} \mid s \succ t\}$.

Theorem 4.43 (Reduction Pair processor)**.** *Let $\tau = \langle \mathcal{P}, \mathcal{R}, f\rangle_\rightsquigarrow$ be an NDP problem, $(\succeq, \succ)$ be a reduction pair, and π be an argument filtering such that $\pi(\mathcal{P})$ and $\pi(\mathcal{R})$ are TRSs (i.e., satisfy the variable condition).*

The following processor `Proc` *is sound, where* `Proc`(τ) *is defined as*

- $\{\langle \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f\rangle_\rightsquigarrow, \langle \mathcal{P} \setminus \mathcal{P}_\pi, \mathcal{R}, f\rangle_\rightsquigarrow\}$ *if the following conditions hold:*
  1. $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succeq_\pi} = \mathcal{P}$
  2. $\mathcal{R}_{\succeq_\pi} = \mathcal{R}$
  3. $\exists \mathcal{P}_\pi \subseteq \mathcal{P}$ *s.t.* $\mathcal{P}_\pi \neq \emptyset$*, and* $\pi(r)$ *is ground for every* $l \to r \in \mathcal{P}_\pi$
- $\{\langle \mathcal{P}, \mathcal{R}, f\rangle_\rightsquigarrow\}$ *otherwise.*

Proof. Let $\tau = \langle \mathcal{P}, \mathcal{R}, f\rangle_\rightsquigarrow$ be an infinite NDP problem, and suppose that all the conditions hold and `Proc`$(\tau) = \{\tau_{\not\succ_\pi}, \tau_{\not\pi}\}$, where $\tau_{\not\succ_\pi} = \langle \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f\rangle_\rightsquigarrow$ and $\tau_{\not\pi} = \langle \mathcal{P} \setminus \mathcal{P}_\pi, \mathcal{R}, f\rangle_\rightsquigarrow$ are the two subproblems defined as above.

Since τ is an infinite problem, there must be an infinite chain $\mathcal{Q} \subseteq \mathcal{P}$. We distinguish two cases:

- If there is some $n \in \mathbb{N}$ such that for every $i \geq n$ the pair $s_i \to t_i \in \mathcal{Q}$ belongs to the non decreasing pairs, i.e. $s_i \to t_i \in \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}$, then this is a suffix $\mathcal{Q}'$ of $\mathcal{Q}$ which is an infinite $\tau_{\not\succ_\pi}$ chain. It follows that $\tau_{\not\succ_\pi}$ is not finite and hence `Proc` is sound for this case.
- Otherwise, there is some $s_i \to t_i \in \mathcal{P}_{\succ_\pi}$ which occurs infinitely often in $\mathcal{Q}$. We further distinguish two cases.
  - $\mathcal{Q} \cap \mathcal{P}_\pi = \emptyset$. Then this is clearly a $\tau_{\not\pi}$ chain, so $\tau_{\not\pi}$ is not finite; `Proc` is sound for this case too.
  - otherwise, there is some pair $s \to t \in \mathcal{P}_\pi$ which occurs at least once in $\mathcal{Q}$. Therefore we can apply Theorem 4.39 to prove that this is a finite narrowing chain, as it follows from [*Arts and Giesl*, 2000, Theorems 7 and 11] that $(\pi(\mathcal{Q}), \pi(\mathcal{R}))$ is a finite rewriting chain.

    Concretely, by stability and monotony of $\succeq_\pi$ and condition (2) above, it follows that $\rightsquigarrow_\mathcal{R}$ is weakly decreasing. From (1) above, stability of $\succ_\pi$, and the fact that $\succ_\pi$ or $\succeq_\pi$ are compatible, we have that $\rightsquigarrow_\mathcal{Q}$ is also weakly decreasing. We have $s_0 \succeq_\pi t_0 \succeq_\pi s_1 \succeq_\pi \cdots s_i \succ_\pi t_i \succeq_\pi \cdots$. The assumption that $s_i \to t_i \in \mathcal{P}_{\succ_\pi}$ occurring infinitely often together with the well-foundedness of $\succ_\pi$ yields that this must be a finite chain, reaching a contradiction, which proves that this case cannot happen. $\square$

The reduction pair processor introduced in our first version of the approach to termination of narrowing with dependency pairs [*Alpuente et al.*, 2008a] could only deal with cycles, where the one presented in Theorem 4.43 can deal with SCCs instead. The original version was also based in Theorem 4.39, but we missed a way to reconcile the requirement of one ground filtered right hand side per cycle with the requirement of decreasingness. This is accomplished in the processor of Theorem 4.39 by returning two problems instead of one as it is usual. To understand why, suppose that we want to prove the finiteness of a problem $\langle\{(1),(2),(3)\},\mathcal{R},\mathbf{g}\rangle_\rightsquigarrow$ where the dependency graph is:

$$\begin{tikzcd}
(1) \arrow[loop left] \arrow[rr] & & (3) \arrow[loop right] \arrow[dl] \\
& (2) \arrow[ul] &
\end{tikzcd}$$

Notice that there are three cycles grouped in a single SCC. For a succesful proof of termination, Theorem 4.25 requires us to prove that there is no infinite chain. There are three possibly infinite chains to consider, one for each cycle. Let $(\succeq_1, \succ_1)$ be a reduction pair and π_1 an argument filtering such that, using the definitions of Theorem 4.43, we get $\mathcal{P}_{\succ_\pi} = \{(1)\}$ and $\mathcal{P}_\pi = \{(3)\}$, and all the other conditions are satisfied as well. From this information, Theorem 4.39 only allows us to conclude that the outermost cycle corresponds to a finite chain. To succesfully show termination we still need to prove finiteness of the chains associated to the two singleton cycles in (1) and (3). This is consistent with the behaviour of our reduction pair processor, which returns the two subproblems

On the other hand, consider a reduction pair $(\succeq_2, \succ_2)$ and an argument filtering π_2 such that, using the definitions of Theorem 4.43, we get $\mathcal{P}_{\succ_\pi} = \{(1)\}$ and $\mathcal{P}_\pi = \{(1)\}$, and all the other conditions are satisfied as well.

Example 4.44. *Recall he NDP problem $\langle\{(1),(3)\},\mathcal{R}_{Policy},\mathbf{m}\rangle_\rightsquigarrow$ resulting from Example 4.42. For convenience, (1) and (3) from Figure 4.2 are displayed below:*

$$\texttt{filter}^\#(\texttt{pckt}(\texttt{194.179.1.}x{:}p, dst, \texttt{new})) \to \texttt{filter}^\#(\texttt{pckt}(\texttt{secure}, dst, \texttt{new})) \quad (1)$$

$$\texttt{filter}^\#(\texttt{pckt}(\texttt{158.42.}x.y{:}p, dst, \texttt{new})) \to \texttt{filter}^\#(\texttt{pckt}(\texttt{secure}, dst, \texttt{new})) \quad (3)$$

We show how this problem can be solved by means of the reduction pair processor of Theorem 4.43. Soundness requires an argument filtering π that makes ground the right-hand side of at least one of the pairs. We can use $\pi(\texttt{pckt}) = [1,3]$ to ground both pairs, with π the identity for any other symbol. The resulting filtered pairs are:

$$\texttt{filter}^\#(\texttt{pckt}(\texttt{194.179.1.}x{:}p, \texttt{new})) \to \texttt{filter}^\#(\texttt{pckt}(\texttt{secure}, \texttt{new})) \quad (1')$$

$$\texttt{filter}^\#(\texttt{pckt}(\texttt{158.42.}x.y{:}p, \texttt{new})) \to \texttt{filter}^\#(\texttt{pckt}(\texttt{secure}, \texttt{new})) \quad (3')$$

and the resulting filtered rules are:

$$
\begin{aligned}
\texttt{filter}(\texttt{pckt}(src, \texttt{established})) &\to \texttt{accept}\\
\texttt{filter}(\texttt{pckt}(\texttt{eth0}, \texttt{new})) &\to \texttt{accept}\\
\texttt{filter}(\texttt{pckt}(\texttt{194.179.1}.x{:}port, \texttt{new})) &\to \texttt{filter}(\texttt{pckt}(\texttt{secure}, \texttt{new}))\\
\texttt{filter}(\texttt{pckt}(\texttt{158.42}.x.y{:}port, \texttt{new})) &\to \texttt{filter}(\texttt{pckt}(\texttt{secure}, \texttt{new}))\\
\texttt{filter}(\texttt{pckt}(\texttt{secure}, \texttt{new})) &\to \texttt{accept}\\
\texttt{filter}(\texttt{pckt}(\texttt{secure}, \texttt{new})) &\to \texttt{drop}\\
\texttt{filter}(\texttt{pckt}(\texttt{ppp0}, \texttt{new})) &\to \texttt{drop}\\
\texttt{filter}(\texttt{pckt}(\texttt{123.123.1.1}\ {:}port, \texttt{new})) &\to \texttt{accept}\\
\texttt{pckt}(\texttt{10.1.1.1}\ {:}port, s) &\to \texttt{pckt}(\texttt{123.23.1.1}\ {:}port, s)\\
\texttt{pckt}(\texttt{10.1.1.2}\ {:}port, s) &\to \texttt{pckt}(\texttt{123.23.1.1}\ {:}port, s)\\
\texttt{pckt}(src, \texttt{new}) &\to \texttt{natroute}(\texttt{pckt}(src, \texttt{established}),\\
&\qquad\qquad\qquad \texttt{pckt}(src, \texttt{established}))\\
\texttt{natroute}(a, b) &\to a\\
\texttt{natroute}(a, b) &\to b
\end{aligned}
$$

Both $\pi(\mathcal{P})$ and $\pi(\mathcal{R}_{Policy})$ satisfy the variable condition, so it is not needed to refine π anymore. We can use an RPO based reduction pair [Arts and Giesl, 2000] to show that both pairs (1) and (3) are strictly decreasing, and enable the reduction pair processor of Theorem 4.43 to obtain two NDP problems with no pairs, which can be shown finite using the dependency graph processor of Theorem 4.41, completing the termination proof.

4.3.3 Usable Rules

In the reduction pair processor presented above, it is required that all the rules in $\mathcal{R}$ are oriented with the non strict part $\succeq_\pi$ of the reduction pair. But it turns out that, given a chain $\mathcal{C} = s_1 \to t_1, s_2 \to t_2 \ldots$, only a subset $\mathcal{U}(\mathcal{C})$ of the rules in $\mathcal{R}$ are used to connect every t_i to s_{i+1}. By taking advantage of this, we can formulate an improved reduction pair processor in which only $\mathcal{R}_{\succeq_\pi} = \mathcal{U}(\mathcal{C})$ is required.

In practice, the exact usable rules $\mathcal{U}(\mathcal{C})$ of a chain $\mathcal{C}$ are uncomputable, and a number of approximations exist in the literature. These approximations differ in the degree of refinement, degree of applicability (some are only applicable to innermost termination, for instance), and the constraints they impose to the DP problem they are applied to. A very accurate rendition of the topic is given in [*Thiemann*, 2007]. Of course this is for termination of the rewriting relation, but it turns out that as Proposition 4.24 suggests, the usable rules of an NDP problem coincide with the usable rules of the corresponding rewriting DP problem.

Below we incorporate the usable rules of [*Giesl et al.*, 2006b] to the reduction pair processor of Theorem 4.43. The impact of the usable rules refinement on the power of this processor is even greater than in the rewriting setting, as in addition of having potentially less ordering constraints which must be satisfied, there are also potentially less rules on which the variable condition needs to be enforced after filtering a right-hand side of $\mathcal{P}$ to make it ground.

In the following, given a TRS $\mathcal{R}$ and symbol $\mathsf{f} \in \mathcal{F}$, we let $\mathsf{Def}_{\mathcal{R}}(\mathsf{f}) = \{l \to r \in \mathcal{R} \mid \mathsf{root}(l) = \mathsf{f}\}$ and $\mathcal{R}_{\langle \mathsf{f} \rangle} = \mathcal{R} \setminus \mathsf{Def}_{\mathcal{R}}(\mathsf{f})$.

We define the usable rules $\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P})$ of an NDP problem $\langle \mathcal{P}, \mathcal{R}, f \rangle_{\rightsquigarrow}$ with regard to an argument filtering π as follows.

Definition 4.45 (Estimated Usable Rules w.r.t. an argument filtering [Giesl et al., 2006b])**.** *For any TRS $\mathcal{R}$ and any argument filtering π, we define*

- $\mathcal{U}^{\pi}_{\mathcal{R}}(x) = \emptyset$ *for* $x \in \mathcal{V}$.
- $\mathcal{U}^{\pi}_{\mathcal{R}}(\mathsf{f}(t_1, \cdots, t_n)) = \mathsf{Def}_{\mathcal{R}}(\mathsf{f}) \cup \bigcup_{l \to r \in \mathsf{Def}_{\mathcal{R}}(\mathsf{f})} \mathcal{U}^{\pi}_{\mathcal{R}_{\langle \mathsf{f} \rangle}}(r) \cup \bigcup_{i \in RegPos_{\pi}(\mathsf{f})} \mathcal{U}^{\pi}_{\mathcal{R}_{\langle \mathsf{f} \rangle}}(t_i)$

For any set of rules $\mathcal{P}$ we define $\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}) = \bigcup_{l \to r \in \mathcal{P}} \mathcal{U}^{\pi}_{\mathcal{R}}(r)$.

In addition, we introduce the notion of $\mathcal{C}_{\mathcal{E}}$-compatibility of a reduction pair. Most of the reduction pairs used in practice are $\mathcal{C}_{\mathcal{E}}$-compatible.

Definition 4.46 ($\mathcal{C}_{\mathcal{E}}$)**.** [Gramlich, 1994, Def.21] *We let $\mathcal{C}_{\mathcal{E}}$ denote the TRS*

$$\{\mathsf{c}_{\mathcal{E}}(x, y) \to x; \mathsf{c}_{\mathcal{E}}(x, y) \to y\}$$

where $\mathsf{c}_{\mathcal{E}}$ is a fresh function symbol.

A TRS $\mathcal{R}$ is said to be $\mathcal{C}_{\mathcal{E}}$-terminating if $\mathcal{R} \cup \mathcal{C}_{\mathcal{E}}$ is terminating.

A relation $\succeq$ is said to be $\mathcal{C}_{\mathcal{E}}$-compatible iff $\mathsf{c}_{\mathcal{E}}(x, y) \succeq x$ and $\mathsf{c}_{\mathcal{E}}(x, y) \succeq y$.

A reduction pair $(\succeq, \succ)$ is said to be $\mathcal{C}_{\mathcal{E}}$-compatible iff $\succeq$ is $\mathcal{C}_{\mathcal{E}}$-compatible.

Below we define the improved reduction pair processor with usable rules and prove its soundness.

Theorem 4.47 (Reduction Pair processor with usable rules)**.** *Let $\tau = \langle \mathcal{P}, \mathcal{R}, f \rangle_{\rightsquigarrow}$ be an NDP problem, $(\succeq, \succ)$ be a $\mathcal{C}_{\mathcal{E}}$-compatible reduction pair and π be an argument filtering such that $\pi(\mathcal{P})$ and $\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}))$ are TRSs. The following processor* `Proc` *is sound, where* `Proc`(τ) *is defined as*

- $\{\langle \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f \rangle_{\rightsquigarrow}, \langle \mathcal{P} \setminus \mathcal{P}_{\pi}, \mathcal{R}, f \rangle_{\rightsquigarrow}\}$ *if the following conditions hold:*
  1. $f = \mathbf{m}$
  2. $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succeq_\pi} = \mathcal{P}$
  3. $\mathcal{R}_{\succeq_\pi} = \mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P})$
  4. $\exists \mathcal{P}_{\pi} \subseteq \mathcal{P}$, $\mathcal{P}_{\pi} \neq \emptyset$, *and $\pi(r)$ is ground for every $l \to r \in \mathcal{P}_{\pi}$*
- $\{\langle \mathcal{P}, \mathcal{R}, f \rangle_{\rightsquigarrow}\}$ *otherwise.*

Before giving the proof, we need to extend the simulation between narrowing chains and filtered rewriting chains given in Theorem 4.39 to enforce the variable condition only upon the usable rules.

Theorem 4.48. *Let $\mathcal{R}$ be a GTRS over a signature Σ, $\mathcal{P}$ be a GTRS over a signature $\Sigma^{\#}$, and π an AF over $\Sigma^{\#}$ s.t. $\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}))$ and $\pi(\mathcal{P})$ are TRSs, and $\pi(t)$ is ground for at least one pair $s \to t \in \mathcal{P}$. If $(\mathcal{P}, \mathcal{R})$ is an infinite minimal narrowing chain , then $(\pi(\mathcal{P}), \pi(\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P})) \cup \mathcal{C}_{\mathcal{E}})$ is an infinite rewriting chain.*

Proof. We prove the theorem by means of a simulation between the narrowing derivation and the filtered rewriting derivation. We assume that $\pi(r_1)$ is ground for the first pair $l_1 \to r_1 \in \mathcal{P}$. There is no loss of generality as we can always skip a prefix of $\mathcal{P}$ and still have an infinite narrowing chain.

Given an infinite minimal $(\mathcal{P}$,$\mathcal{R})$–narrowing chain, by definition there is an infinite $\rightsquigarrow_{\mathcal{R}\cup\mathcal{P}}$ derivation with an infinite number of $\mathcal{P}$ steps of the form:

$$l_1\sigma_1 \overset{\varepsilon}{\rightsquigarrow}_{\sigma_1,l_1\to r_1} r_1\sigma_1 \overset{>\epsilon *}{\rightsquigarrow}_{\tau_1,\mathcal{R}} t_2 \overset{\varepsilon}{\rightsquigarrow}_{\sigma_2,l_2\to r_2} u_2 \overset{>\epsilon *}{\rightsquigarrow}_{\tau_2,\mathcal{R}} \cdots$$

By the soundness of narrowing, this is also an infinite rewriting derivation of the form:

$$l_1\theta \overset{\epsilon}{\to}_{l_1\to r_1} r_1\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} t_2\theta \overset{\epsilon}{\to}_{l_2\to r_2} u_2\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} \cdots$$

where θ is a substitution possibly carrying infinite terms.

By claims (ii) and (v) of [*Thiemann*, 2007, Lemma 4.35], we know that the derivation above implies that there exists the following filtered derivation:

$$\pi(l_1)\mathcal{I}_\pi(\theta) \overset{\epsilon}{\to}_{\pi(l_1)\to\pi(r_1)} \pi(r_1)\mathcal{I}_\pi(\theta) \overset{>\epsilon *}{\to}_{\pi(\mathcal{U}^\pi_\mathcal{R}(\mathcal{P}))\cup\mathcal{C}_\mathcal{E}} \pi(t_2)\mathcal{I}_\pi(\theta) \overset{\epsilon}{\to}_{\pi(l_2)\to\pi(r_2)} u_2\mathcal{I}_\pi(\theta)\cdots$$

where $\mathcal{I}_\pi$ refers to the transformation in [*Thiemann*, 2007, Definition 4.34]. $\mathcal{I}_\pi(\theta)$ is only computable when the terms in θ are finite and terminating. However this is irrelevant here since, as $\pi(r_1)$ is ground and $\pi(\mathcal{R})$ has no extra variables, this is a fully ground derivation where the $\mathcal{I}_\pi(\theta)$ substitutions are ignored and we have only finite terms. Thus we have:

$$\pi(l_1)\mathcal{I}_\pi(\theta) \overset{\epsilon}{\to}_{\pi(l_1)\to\pi(r_1)} \pi(r_1) \overset{>\epsilon *}{\to}_{\pi(\mathcal{U}^\pi_\mathcal{R}(\mathcal{P}))\cup\mathcal{C}_\mathcal{E}} \pi(t_2) \overset{\varepsilon}{\to}_{\pi(l_2)\to\pi(r_2)} u_2 \overset{>\epsilon *}{\to}_{\pi(\mathcal{U}^\pi_\mathcal{R}(\mathcal{P}))\cup\mathcal{C}_\mathcal{E}} \cdots$$

From this infinite derivation, it follows that the sequence of pairs $\pi(l_1) \to \pi(r_1)$, $\pi(l_2) \to \pi(r_2), \ldots$, constitutes a $(\pi(\mathcal{P}),\mathcal{U}^\pi_\mathcal{R}(\mathcal{P}) \cup \mathcal{C}_\mathcal{E})$ rewriting chain. As $\mathcal{P}$ is infinite, there are an infinite number of $\pi(\mathcal{P})$ steps in this filtered derivation, and we are done. □

We proceed now with the proof of Theorem 4.47.

Proof of Theorem 4.47. The proof follows the scheme of the proof of Theorem 4.43, using the simulation for usable rules given in Theorem 4.48 instead of the plain one of Theorem 4.39.

Let $\tau = \langle \mathcal{P}, \mathcal{R}, \mathbf{m}\rangle_\rightsquigarrow$ be a non finite NDP problem, and suppose that all the conditions hold and $\texttt{Proc}(\tau) = \{\tau_{\not\succ_\pi}, \tau_{\not\gtrsim}\}$, where $\tau_{\not\succ_\pi} = \langle \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f\rangle_\rightsquigarrow$ and $\tau_{\not\gtrsim} = \langle \mathcal{P} \setminus \mathcal{P}_\pi, \mathcal{R}, f\rangle_\rightsquigarrow$ are the two subproblems.

Since τ is not finite, there must be an infinite chain $\mathcal{Q} \subseteq \mathcal{P}$. We distinguish two cases:

- If there is some $n \in \mathbb{N}$ such for every $i \geq n$ the pair $s_i \to t_i \in \mathcal{Q}$ belongs to the non decreasing pairs, i.e. $s_i \to t_i \in \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}$, then this is clearly a $\tau_{\not\succ_\pi}$ chain and we are done.

- Otherwise, there is some $s_i \to t_i \in \mathcal{P}_{\succ_\pi}$ which occurs infinitely often in $\mathcal{Q}$. We further distinguish two cases.
  - $\mathcal{Q} \cap \mathcal{P}_\pi = \emptyset$. Then this is clearly a $\tau_{\not\pi}$ chain and we are done.
  - Otherwise, there is some pair $s \to t \in \mathcal{P}_\pi$ which occurs at least once in the chain. We proceed now by contradiction. Assume that $\mathcal{Q}$ is an infinite narrowing chain. By Theorem 4.48, we know that $(\pi(\mathcal{Q}), \pi(\mathcal{U}^\pi_\mathcal{R}(\mathcal{Q})))$ must be an infinite rewriting chain. However this contradicts the fact that from the assumptions we can infer by [Arts and Giesl, 2000, Theorems 7 and 11] that $(\pi(\mathcal{Q}), \pi(\mathcal{U}^\pi_\mathcal{R}(\mathcal{Q})))$ is actually a finite rewriting chain. □

Example 4.49. *Consider the TRS given by the following set of rules:*

$$\mathtt{f}(\mathtt{a}(x), x) \to \mathtt{f}(\mathtt{b}(x), x) \tag{4.6}$$
$$\mathtt{b}(x) \to \mathtt{c} \tag{4.7}$$
$$\mathtt{h}(\mathtt{b}(x)) \to x \tag{4.8}$$

Narrowing is terminating in this TRS, as it can be proven with the reduction pair with usable rules processor of Theorem 4.47. On the other hand the reduction pairs processor of Theorem 4.43 does not lead to a succesful termination proof. We obtain one ll–dependency pair (4.11) and two lr–dependency pairs (4.9) and (4.10).

$$\mathtt{F}(\mathtt{a}(x), x) \to \mathtt{F}(\mathtt{b}(x), x) \tag{4.9}$$
$$\mathtt{F}(\mathtt{a}(x), x) \to \mathtt{B}(x) \tag{4.10}$$
$$\mathtt{H}(\bullet(x)) \to \mathtt{B}(x) \tag{4.11}$$

The associated dependency graph contains only one SCC with a single pair {(4.9)}. Therefore there is only one NDP problem to solve, which is:

$$\left\langle \mathtt{F}(\mathtt{a}(x), x) \to \mathtt{F}(\mathtt{b}(x), x),\ \begin{array}{ll} \mathtt{f}(\mathtt{a}(x), x) & \to \mathtt{f}(\mathtt{b}(x), x) \\ \mathtt{b}(x) & \to \mathtt{c} \\ \mathtt{h}(\mathtt{b}(x)) & \to x \end{array},\ \mathbf{m} \right\rangle_{\leadsto}$$

Let's try first with the reduction pair processor of Theorem 4.43. In order to satisfy condition 3, we need to synthetize an argument filtering π such that $\pi(\mathtt{F}(\mathtt{b}(x), x))$ is ground. There are only two options. Filtering the two arguments of F *leads to the filtered pair* $\mathtt{F} \to \mathtt{F}$ *which is clearly non-decreasing. Instead one can filter the second argument of* F *and the argument of* b*, using $\pi(\mathtt{F}) = [1], \pi(\mathtt{b}) = []$, and π the identity for any other symbol. The resulting filtering of the NDP problem is:*

$$\left\langle \mathtt{F}(\mathtt{a}(x)) \to \mathtt{F}(\mathtt{b}),\ \begin{array}{rl} \mathtt{f}(\mathtt{a}(x), x) & \to \mathtt{f}(\mathtt{b}, x) \\ \mathtt{b} & \to \mathtt{c} \\ \mathtt{h}(\mathtt{b}) & \to x \end{array},\ \mathbf{m} \right\rangle_{\leadsto}$$

In order to enforce the variable condition on $\mathcal{P}$, we would need to fix the extra variable in the rule $\mathtt{h}(\mathtt{b}) \to x$, but there is no refinement of π which can filter the extra variable *in the right-hand side. Therefore the processor of Theorem 4.43 cannot be applied to this problem.*

On the other hand if we consider the reduction pairs processor extended with usable rules, we can use the same filtering π as above to satisfy condition 4, which induces a single usable rule $\mathtt{b}(x) \to \mathtt{c}$. The resulting filtered problem is: $\langle \mathtt{F}(\mathtt{a}(x)) \to \mathtt{F}(\mathtt{b}), \mathtt{b} \to \mathtt{c}, \mathbf{m}\rangle_{\rightsquigarrow}$ which can be oriented using a RPO reduction pair with precedence $\mathtt{a} > \mathtt{b} > \mathtt{c}$, succesfully obtaining a termination proof.

4.3.4 The Argument Filtering processor

We claim that it is possible to adapt most of the standard DP processors in order to deal with the grounding AF requirement. But on the other hand, our next processor can transform an NDP problem into an ordinary one. Afterwards, any existing DP processor for rewriting becomes applicable.

Theorem 4.50 (Argument Filtering Processor)**.** *Let $\tau = \langle \mathcal{P}, \mathcal{R}, f\rangle_{\rightsquigarrow}$ be an NDP problem and π be an argument filtering s.t. both $\pi(\mathcal{P})$ and $\pi(\mathcal{R})$ are TRSs, and $\pi(t)$ is ground for at least one pair $s \to t \in \pi(\mathcal{P})$. Then, $\mathtt{Proc}(\tau)$ returns $\{\tau_{\to}, \tau_{\not\pi}\}$, where:*

- *$\mathcal{P}_\pi \subseteq \mathcal{P}$, $\mathcal{P}_\pi \neq \emptyset$, and $\pi(r)$ is ground for every $l \to r \in \mathcal{P}_\pi$*
- *$\mathcal{P}_\rhd = \{l \to r \mid l \to r \in \mathcal{P}, \pi(l) \rhd \pi(r)\}$*
- *$\tau_{\not\pi} = \langle \mathcal{P} \setminus \mathcal{P}_\pi, \mathcal{R}, f\rangle_{\rightsquigarrow}$ is an NDP problem,*
- *$\tau_{\to} = \langle \pi(\mathcal{P} \setminus \mathcal{P}_\rhd), \pi(\mathcal{R}), \mathbf{a}\rangle$ is a rewriting DP problem,*

$\mathtt{Proc}$ is a sound NDP processor.

Proof. Suppose that τ is not finite. Then there is an infinite narrowing chain $(\mathcal{Q}, \mathcal{R})$ with $\mathcal{Q} \subseteq \mathcal{P}$. We distinguish two cases.

- $\mathcal{Q} \cap \mathcal{P}_\pi = \emptyset$. Then this is clearly a $\tau_{\not\pi}$ chain, and we are done.
- Otherwise, there is some pair from $\mathcal{P}_\pi$ which occurs infinitely often in $\mathcal{Q}$. By Theorem 4.39, there is an infinite $(\pi(\mathcal{Q}), \pi(\mathcal{R}))$-rewriting chain. As the set of discarded pairs $\pi(\mathcal{P}_\rhd)$ cannot produce infinite rewriting chains [*Dershowitz*, 2003], this must be an infinite $(\pi(\mathcal{P} \setminus \mathcal{P}_\rhd), \pi(\mathcal{R}))$-rewriting chain. Therefore the problem $\tau_{\to}$ is not finite, and we are done. □

Example 4.51. *In example 4.44, we showed how to solve the NDP problem $\langle\{(1), (3)\}, \mathcal{R}_{Policy}, \mathbf{m}\rangle_{\rightsquigarrow}$ of Example 4.42, using the reduction pair processor. Now we show how the AF processor and a tool for solving termination of rewriting (DP) problems can be used for the same purpose.*

As for the reduction pair processor, soundness requires an argument filtering π which makes ground the right-hand side of at least one of the pairs. We can use the same π as in example 4.44, i.e. $\pi(\mathtt{pckt}) = [1, 3]$, and the identity for all other symbols. As seen in Example 4.44, this argument filtering already enforces the variable *condition on both the rules and the pairs of the problem. We can apply the argument filtering processor of Theorem 4.50, obtaining the NDP problem $\langle\{\}, \mathcal{R}_{Policy}, \mathbf{m}\rangle_{\rightsquigarrow}$ and the DP problem $\langle\pi(\mathcal{P}), \pi(\mathcal{R}_{Policy}), \mathbf{a})\rangle$. The former is trivially finite by the Dependency Graph processor of Theorem 4.41. The latter can be solved by any modern termination tool implementing the DP method, such as Aprove [Giesl et al., 2004], Mu-Term [Alarcón et al., 2007], or the small, RPO based reduction pair solver in our tool [Narradar].*

The argument filtering processor of Theorem 4.50 cannot be used to solve Example 4.49, where the variable condition on the rules precludes the use of an argument filtering which leads to a successful proof. In this sense, the argument filtering processor exhibits the same weaknesses as the first reduction pair processor introduced in Theorem 4.43. For this reason we introduce a new version equipped with a notion of usable rules, analogously to what we did in Theorem 4.47 for the reduction pair processor.

Theorem 4.52 (Argument filtering with usable rules)**.** *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathbf{m})$ be a narrowing DP problem, $\mathcal{P}_1 \dots \mathcal{P}_n$ be the sets of nodes corresponding to the SCCs in its associated estimated dependency graph, and π be an argument filtering s.t. $\pi(\mathcal{P})$ and $\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}))$ are TRSs, and $\pi(t)$ is ground for at least one pair $s \to t \in \pi(\mathcal{P})$. Then, $Proc(\tau) = \{\tau_{\not\pi}, \tau_{\to}\}$, where:*

- $\mathcal{P}_\pi \subseteq \mathcal{P}$*,* $\mathcal{P}_\pi \neq \emptyset$*, and* $\pi(r)$ *is ground for every* $l \to r \in \mathcal{P}_\pi$
- $\mathcal{P}_\rhd = \{l \to r \mid l \to r \in \mathcal{P}, \pi(l) \rhd \pi(r)\}$
- $\tau_{\not\pi} = \langle\mathcal{P} \setminus \mathcal{P}_\pi, \mathcal{R}, f\rangle_{\rightsquigarrow}$ *is an NDP problem,*
- $\tau_{\to} = \langle\pi(\mathcal{P} \setminus \mathcal{P}_\rhd), \mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{C}_{\mathcal{E}}, \mathbf{a}\rangle$ *is a DP problem,*

Proc is a sound narrowing DP processor.

Proof. Identical to the proof of Theorem 4.50, but using Lemma 4.48 instead of Lemma 4.39.

Suppose that τ is not finite. Then there is an infinite narrowing chain $(\mathcal{Q}, \mathcal{R})$ with $\mathcal{Q} \subseteq \mathcal{P}$. We distinguish two cases.

- $\mathcal{Q} \cap \mathcal{P}_\pi = \emptyset$. Then this is clearly also a chain in $\tau_{\not\pi}$ and we are done.
- Otherwise, there is some pair from $\mathcal{P}_\pi$ which occurs infinitely often in $\mathcal{Q}$. By Theorem 4.48, there is an infinite $(\pi(\mathcal{P}), \mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{C}_{\mathcal{E}})$ rewriting chain. This must be in fact a $(\pi(\mathcal{P} \setminus \mathcal{P}_\rhd), \mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}) \cup \mathcal{C}_{\mathcal{E}})$ chain, since the pairs $\pi(\mathcal{P}_\rhd)$ cannot produce infinite rewriting chains [*Dershowitz*, 2003]. Therefore the problem $\tau_{\to}$ is not finite and we are done. □

With the version of the argument filtering processor with usable rules, we are now equipped to solve Example 4.49.

Example 4.53. *Consider the NDP problem of Example 4.49:*

$$\left\langle \mathtt{F}(\mathtt{a}(x),x) \to \mathtt{F}(\mathtt{b}(x),x),\quad \begin{array}{rl} \mathtt{f}(\mathtt{a}(x),x) & \to \mathtt{f}(\mathtt{b}(x),x) \\ \mathtt{b}(x) & \to \mathtt{c} \\ \mathtt{h}(\mathtt{b}(x)) & \to x \end{array}, \mathbf{m} \right\rangle_{\leadsto}$$

Let's try first with the reduction pair processor of Theorem 4.43. In order to satisfy condition 3, we need to synthetize an argument filtering π such that $\pi(\mathtt{F}(\mathtt{b}(x),x))$ is ground. There are only two options. Filtering the two arguments of F *leads to the filtered pair* $\mathtt{F} \to \mathtt{F}$ *which is clearly non-decreasing. Instead, we can use the same π employed in Example 4.49, $\pi(\mathtt{F}) = [1], \pi(\mathtt{b}) = []$, and the identity for any other symbol, which induces a single usable rule $\mathtt{b}(x) \to \mathtt{c}$. Applying the argument filtering processor of Theorem 4.52, we obtain an NDP problem:*

$$\left\langle \{\},\quad \begin{array}{rl} \mathtt{f}(\mathtt{a}(x),x) & \to \mathtt{f}(\mathtt{b}(x),x) \\ \mathtt{b}(x) & \to \mathtt{c} \\ \mathtt{h}(\mathtt{b}(x)) & \to x \end{array}, \mathbf{m} \right\rangle_{\leadsto}$$

and a DP problem

$$\left\langle \mathtt{F}(\mathtt{a}(x)) \to \mathtt{F}(\mathtt{b}),\quad \begin{array}{rl} \mathtt{b} & \to \mathtt{c} \\ \mathtt{c}_{\mathcal{E}}(x,y) & \to x \\ \mathtt{c}_{\mathcal{E}}(x,y) & \to y \end{array}, \mathbf{a} \right\rangle$$

The NDP problem is trivially finite by the dependency graph processor of Theorem 4.41, and the DP problem can be trivially proven finite by any modern DP framework tool.

4.4 Discussion

In the previous section we generalized the DP framework to narrowing by defining the notions of NDP problem and processor, and adapted a number of existing DP processors, including the dependency graph processor and the reduction pair (with usable rules) processor. Finally, we gave a processor, namely the argument filtering (with usable rules) processor which can transform an NDP problem into a DP problem, enabling the use of any existing DP processor to prove its finiteness.

By means of the pairs of Examples (4.44, 4.51) and (4.49, 4.53) we hope to have illustrated the duality of

1. adapting an existing DP processor $\mathcal{P}$ to the NDP framework, and
2. converting an NDP problem to a DP problem and then solving it with $\mathcal{P}$.

While these two approaches have comparable power (although we do not provide any formal proof of this statement), there are significant tradeoffs made in each case.

By choosing to transform an NDP problem into a DP problem it is possible to directly reuse existing tools without modifying them. While this is obviously very compelling, the problem of finding the argument filtering which enables this step is inherently incomplete and can only be solved by search; it cannot be cast as an optimization problem. Moreover, it has an exponential complexity in the sums of the arities of the symbols in the signature. In other words, it is terribly inefficient. In practice, others have approached this problem by using heuristics [*Nishida and Vidal*, 2010] to find an argument filtering. Our position is that adapting existing DP processors should be preferred over the argument filtering processors of Theorems 4.50 and 4.52.

By choosing to adapt existing DP processors to the NDP framework, an implementation effort needs to be made. We argue that this effort is very lightweight in most cases, and this is the approach that we have followed in our own tool, the termination prover [Narradar]. NARRADAR offers a web interface accepting TRSs in the standard TPDB format from the termination problem database[11], with a few extensions to denote narrowing and initial goal problems, as well as logic programs.

The techniques of this chapter by themselves take roughly 500 lines of code. In particular, NARRADAR provides an RPO based reduction pair implemented by encoding the order restrictions in propositional form and then using the [Yices] SAT solver to find the precedence and argument filtering which orient the rules and pairs. This approach is described in detail in [*Codish et al.*, 2005; *Schneider-Kamp et al.*, 2007; *Codish et al.*, 2006]. NARRADAR extends this approach with two additional constraint in the propositions generated for narrowing problems, which ensure that:

- The argument filtering enforces the variable condition in the rules and pairs.
- At least one right-hand side of a pair is ground when filtered.

With the SAT encoding in place, these additional constraints together amounted to less than 10 lines of Haskell. We claim that other reduction pairs can be similarly adapted with moderate effort by including these two additional constraints. Additionally, NARRADAR implements all the processors described in this Chapter including the notion of usable rules. The tool is capable of proving termination of narrowing in e.g. the leading example of this chapter in less than a second.

We have not considered here the adaptation of other kinds of DP processors; there are too many to fit in this chapter. But given the central role that argument filterings play in proofs of termination of narrowing, we expect that graph refinement processors, such as the instantiation and narrowing processors of [*Arts and Giesl*, 2000] will prove very useful in the NDP framework. These processors can still be applied after transforming to the DP framework, but by then the argument filtering may have eliminated too much information from the problem.

[11]http://www.lri.fr/ marche/tpdb/format.html

5 Automatic Termination from an Initial Goal

In the previous section we discussed a method to automatically prove the termination of narrowing in a TRS, for any initial query. In some cases, however, one is only interested in those narrowing sequences which start from a term or a set of terms. This chapter proposes a method for proving the termination of narrowing derivations which stem from an initial set of terms. The method is built on top of two extensions to the dependency pair framework. We consider these extensions individually, before combining them to prove termination of narrowing. Moreover, the applicability of termination of narrowing from an initial goal is showcased by considering proofs of termination of logic programs obtained via termination of narrowing, and the effectiveness of the technique developed is demonstrated by an empirical analysis before concluding.
Concretely, the chapter contains the following three contributions.

The problem of proving finiteness of the derivations that start from a distinguished set of terms has already been considered in some previous works, e.g., for proving the termination of logic programs [*Schneider-Kamp et al.*, 2009b], for proving the termination of Haskell programs [*Giesl et al.*, 2006a], and for proving the termination of *narrowing* [*Vidal*, 2008]. Unfortunately, these works only consider termination from an initial goal as a by-product, and their results are ad-hoc and difficult to generalize. As a first contribution of this chapter, in Section 5.1 we extend the dependency pair framework in order to consider only derivations from a given initial set of terms. The fundamental improvements are twofold: firstly, we introduce a notion of chain that considers only reachable loops, thus reducing the number of pairs to consider; secondly, we also present a notion of usable rules that regards as usable only those rules occurring in the derivation from an initial goal, allowing us to reduce the number of rules. In addition, our developments are generally useful in the sense that they can be reused for any variant of the DP framework, including termination of rewriting, the variant for termination of Haskell programs considered in [*Giesl et al.*, 2006a], or the variant for termination of logic programs by translating them to rewrite systems of [*Schneider-Kamp et al.*, 2009b]. Most importantly, they are reused in this chapter to study the termination of narrowing from an initial set of terms.

As a second contribution of this chapter, in Section 5.2 we study a direct application of the dependency pair technique to decide relative termination problems. Roughly speaking, $\mathcal{R}$ terminates relative to $\mathcal{B}$ if all $\rightarrow_{\mathcal{R}} \cup \rightarrow_{\mathcal{B}}$ reductions contain only finitely many $\rightarrow_{\mathcal{R}}$ steps. Because in essence the dependency pair approach is a method for deciding the termination of a TRS $\mathcal{R}$ via the relative termination of the TRS $DP(\mathcal{R})$ w.r.t $\mathcal{R}$, it is natural to ask the question of whether the dependency pair approach can be reused to directly decide relative termination of $\mathcal{R}$ w.r.t $\mathcal{B}$ via a dependency pair problem $(\mathcal{R}', \mathcal{B}')$ for some $\mathcal{B}'$ and $\mathcal{R}'$ constructed in terms of the original problem. [*Zantema*, 2004] points out that this is not possible in general. We ascertain a set of sufficient conditions under which this can be done. Namely, relative termination of $\mathcal{R}$ w.r.t. $\mathcal{B}$ can be decided via the problem $(DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$ when, roughly speaking, $\mathcal{B}$ does not make calls to functions defined in $\mathcal{R}$.

As the third and last contribution, the two previous contributions are combined in Section 5.3 to develop a method for proving the termination of narrowing derivations from a set of initial goals. In order to do so, we recast termination of narrowing as a problem of relative termination of the original system w.r.t a set of generators, following the approach of [*Vidal*, 2008]. The results obtained are strictly more general than [*Vidal*, 2008; *Nishida and Vidal*, 2010] which are limited to left-linear systems. They are also strictly more effective, since we replace heuristic-guided search by constraint solving. Finally, they are also potentially more accurate when combined with the framework of the previous chapter.

Termination of Logic Programs

One notable application of the results in this chapter is proving the termination of *logic* programs. In [*Schneider-Kamp et al.*, 2009b], logic programs are transformed into TRSs such that the termination of infinitary constructor rewriting on the resulting TRS implies termination of the original logic program. Unsurprisingly ([*Bosco et al.*, 1988]), narrowing can be used in place of infinitary rewriting with the same purpose, yielding a new method for proving the termination of logic programs. Let us proceed with a brief overview of the method described in [*Schneider-Kamp et al.*, 2009b].

In the following, Δ usually denotes a set of predicate symbols, and $\mathcal{A}(\Sigma, \Delta, \mathcal{V})$ the set of all atoms $p(t_1, \dots, t_n)$, where $p/n \in \Delta$ and $t_1, \dots, t_n \in \mathcal{T}(\Sigma, \mathcal{V})$. A logic program over (Σ,Δ) is as usual a set of clauses h :- $p_0, p_1, \dots, p_n$ for $n \geq 0$, where $h, p_1, \dots, p_n$ are atoms from $\mathcal{A}(\Sigma, \Delta, \mathcal{V})$. A clause is called a *fact* if $n = 0$. A *query* is a set of atoms, and $\Box$ denotes the empty query. We briefly recall the procedural semantics of logic programs based on SLD resolution with the left-to-right selection rule. More details can be found in [*Apt*, 1997], for example.

Definition 5.1 (Logic Derivation)**.** *Let Q be a query $A_1, \dots, A_m$, and c be a clause $H \leftarrow B_1, \dots, B_k$. Q' is a resolvent of Q and c using θ, denoted $Q \vdash_{c,\theta} Q'$ if θ is the most general unifier of A_1 and H, and $Q' = (B_1, \dots, B_k, A_2, \dots, A_m)\theta$.*

Given a program $\mathcal{P}$ and a query Q, a derivation of $\mathcal{P}$ and Q is a possibly infinite sequence of queries $Q_0, Q_1, Q_2, \dots,$ where $Q_0 = Q$ and for all queries we have $Q_i \vdash_{c_{i+1},\theta_{i+1}} Q_{i+1}$ for some substitution θ_{i+1} and fresh variant c_{i+1} of a clause of $\mathcal{P}$.

We often simply write $Q_i \vdash_{\mathcal{P}} Q_{i+1}$.

The query Q terminates for $\mathcal{P}$ if all derivations of $\mathcal{P}$ and Q are finite.

The following is the transformation from logic programs to TRSs defined in [*Schneider-Kamp et al.*, 2009b].

Definition 5.2 (Transformation [*Schneider-Kamp et al.*, 2009b]). *A logic program $\mathcal{P}$ over (Σ,Δ) is transformed into the TRS $\mathcal{R}_{\mathcal{P}}$ over $\Sigma_{\mathcal{P}}$.*

$$\Sigma_{\mathcal{P}} = \Sigma \cup \{p/_{in}, p_{out} \mid p/n \in \Delta\} \cup \bigcup \{u_{c,1} \ldots u_{c,k} \mid c = h \text{ :- } p_1 \ldots p_k \in \mathcal{P}\}$$

The rules of $\mathcal{R}_{\mathcal{P}}$ are constructed as follows:

- *For each fact $p(\overline{a}) \in \mathcal{P}$, $\mathcal{R}_{\mathcal{P}}$ contains the rule $p_{in}(\overline{a}) \to p_{out}(\overline{a})$*
- *For each clause c of the form $p(\overline{a})$:- $p_1(\overline{a_1}), \ldots, p_k(\overline{a_k}) \in \mathcal{P}$, $\mathcal{R}_{\mathcal{P}}$ contains the rules:*

$$\begin{aligned} p_{in}(\overline{a}) &\to u_{c,1}(p_{1_{in}}(\overline{a_1}), \mathit{Var}(\overline{a})) \\ u_{c,1}(p_{1_{out}}(\overline{a_1}), \mathit{Var}(\overline{a})) &\to u_{c,2}(p_{2_{in}}(\overline{a_2}), \mathit{Var}(\overline{a}) \cup \mathit{Var}(\overline{a_1})) \\ &\ldots \\ u_{c,k}(p_{k_{out}}(\overline{a_k}), \mathit{Var}(\overline{a}) \cup \mathit{Var}(\overline{a_1}) \cup \ldots \cup \mathit{Var}(\overline{a_{k-1}})) &\to p_{out}(\overline{a}) \end{aligned}$$

[*Schneider-Kamp et al.*, 2009b] shows how to prove termination of a query Q over a logic program $\mathcal{P}$ in terms of the termination, from an initial set of terms derived from Q, of a special rewriting relation over the TRS $\mathcal{R}_{\mathcal{P}}$ called infinitary constructor rewriting.

Definition 5.3 (Infinitary Constructor Rewriting [*Schneider-Kamp et al.*, 2009b]). *Let $\mathcal{T}^{\infty}(\Sigma, \mathcal{V})$ denote the set of all possibly infinite terms over a signature $\Sigma = \mathcal{C} \uplus \mathcal{D}$ and a set of variables $\mathcal{V}$. The infinitary constructor rewriting relation $\overset{\infty}{\to}_{\mathcal{R}}$ is a binary relation on terms of $\mathcal{T}^{\infty}(\Sigma, \mathcal{V})$ defined by $\overset{\infty}{\to}_{\mathcal{R}} = (C[l\sigma]_p, C[r\sigma]_p)$ where C is a context, p a position, $\sigma : \mathcal{V} \to \mathcal{T}^{\infty}(\mathcal{C}, \mathcal{V})$ an infinitary constructor substitution, and $\mathcal{R}$ a generalized TRS with $l \to r \in \mathcal{R}$.*

Infinitary constructor rewriting restricts substitutions to be constructor, but on the other hand it permits infinite terms. At the end of the chapter we will show that given a logic program $\mathcal{P}$, termination of narrowing over the transformed system $\mathcal{R}_{\mathcal{P}}$ implies the termination of the logic program $\mathcal{P}$.

Example 5.4. *Consider the logic program $\mathcal{P}_{inc}$, extracted from [Schneider-Kamp et al., 2009b, Section 7.2], defined by the clauses:*

```
p(X) :- q(f(Y)), p(Y).
p(g(X)) :- p(X).
q(g(Y)).
```

The transformation of [Schneider-Kamp et al., 2009b] yields the TRS $\mathcal{R}_{inc}$ given by the rules:

$$\begin{aligned}
\mathsf{p_{in}}(\mathsf{g}(X)) &\to \mathsf{u_3}(\mathsf{p_{in}}(X), X)\\
\mathsf{p_{in}}(X) &\to \mathsf{u_1}(\mathsf{q_{in}}(\mathsf{f}(Y)), X)\\
\mathsf{q_{in}}(\mathsf{g}(Y)) &\to \mathsf{q_{out}}(\mathsf{g}(Y))\\
\mathsf{u_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) &\to \mathsf{u_2}(\mathsf{p_{in}}(Y), X, Y)\\
\mathsf{u_2}(\mathsf{p_{out}}(Y), X, Y) &\to \mathsf{p_{out}}(X)\\
\mathsf{u_3}(\mathsf{p_{out}}(X), X) &\to \mathsf{p_{out}}(\mathsf{g}(X))
\end{aligned}$$

Note that $\mathcal{R}_{inc}$ is non-left-linear due to the rules for $\mathsf{u_2}$ and $\mathsf{u_3}$. The method introduced in this chapter can show that narrowing terminates for any derivation starting from a term of the form $\mathsf{p_{in}}(x)$, which in turn, as we will see, implies the termination of the logic program for all queries of the form $\mathsf{p}(<\texttt{term}>)$ for any ground term.

We note that the TRSs generated by the transformation of [*Schneider-Kamp et al.*, 2009b], as seen in the example above, are non-left-linear and often contain extra variables, which means that termination of narrowing from an initial goal cannot be proven with any of the previously existing methods [*Vidal*, 2008; *Nishida and Vidal*, 2010].

Structure of the chapter

We present the goal-directed extension of the dependency pair framework in Section 5.1. First we introduce a specific notion of chain, and then build on top of it to derive the practical notions of goal-directed dependency graph and goal-directed usable-rules. For each of them, we give an algorithm to derive the goal-directed version starting from a standard version. For example, one can reuse the estimations of dependency graph and usable rules introduced in the previous chapter to derive goal-directed versions. Section 5.2 considers the problem of deciding relative termination via the dependency pair approach when a sufficient set of conditions is satisfied. Namely, when the involved TRSs form a hierarchical combination. In fact the hierarchical combination restriction can be slightly relaxed, as seen in Section 5.2.1.

Termination of narrowing is not considered until Section 5.3. First we introduce $\textsc{gen}(\mathcal{R})$, the generator of a system $\mathcal{R}$, a set of rules which can generate every term in $\mathcal{T}(\mathcal{C},\mathcal{V})$. Then we show that $\to_{\mathcal{R}\cup\textsc{gen}(\mathcal{R})}$ in left-linear systems (possibly with extra variables) simulates $\leadsto_{\mathcal{R}}$. Hence, relative termination of $\to_{\mathcal{R}}$ w.r.t. $\to_{\textsc{gen}(\mathcal{R})}$ implies termination of $\leadsto_{\mathcal{R}}$. Moreover, in Section 5.3.1, we lift the left-linear restriction by extending $\textsc{gen}(\mathcal{R})$ to generate normal forms. Since the resulting problems contains a potentially infinite number of rules, normal dependency pair processors cannot be applied to them. In Section 5.3.2 we consider this problem and develop specific versions of the dependency graph processor and the reduction pair processor which can be applied to these problems. Termination of logic programs is briefly developed in

Section 5.4. In short, we demonstrate the soundness of using termination of narrowing in place of infinitary constructor rewriting to prove termination of logic programs in the framework of [*Schneider-Kamp et al.*, 2009b], The chapter ends with an extended empirical evaluation in Section 5.5.

Part of the results in this chapter have been published in [*Iborra et al.*, 2010]. In this chapter we additionally:

- (Section 5.2.1) extend the relative termination criterion to *relaxed* hierarchical combinations, a generalization of hierarchical combinations. This is required for the next contribution,
- (Sections 5.3.1 and 5.3.2) lift the left-linear restriction when proving termination of narrowing from an initial goal, and
- (Section 5.4) provide the foundations for proving termination of logic programs via termination of narrowing.

Since the TRS resulting from a logic program is almost always non-left-linear, the third contribution is only enabled by the two former contributions.

We warn the reader of Chapter 4 that even though the method developed in this chapter and the method developed in there are both based on the Dependency Pair approach, they constitute unrelated instantiations. In particular, the notions of pair and chain are different.

5.1 Goal-directed Dependency Pairs

The particulars of the dependency pair approach are introduced in Section 4.1. In this section, we develop its *goal-directed* extension, where only derivations starting from a given set of terms, denoted by means of an *initial goal*, are considered.

Definition 5.5 (Initial Goal). *Let $\mathcal{R}$ be a TRS over $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ and $t_0 = \mathsf{f}(x_1, \ldots, x_n)$ be a term. We say that t_0 is an initial goal for $\mathcal{R}$ if $\mathsf{f} \in \mathcal{D}$ is a defined function symbol and $x_1, \ldots, x_n \in \mathcal{V}$ are distinct variables.*

Intuitively speaking, an initial goal t_0 represents the set $\lceil t_0 \rceil$ of (non necessarily ground) *constructor* instances of the term t_0, i.e.,

$$\lceil t_0 \rceil \;=\; \{\ t_0\sigma \mid \sigma \text{ is a constructor substitution }\}$$

For instance, given the signature $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$ with $\mathsf{f}/1 \in \mathcal{D}$ and $\mathsf{z}/0, \mathsf{s}/1 \in \mathcal{C}$, the initial goal $\mathsf{f}(x)$ represents the set $\lceil \mathsf{f}(x) \rceil = \{\mathsf{f}(x), \mathsf{f}(\mathsf{z}), \mathsf{f}(\mathsf{s}(x)), \mathsf{f}(\mathsf{s}(\mathsf{z})), \ldots\}$.

It is worthwhile to observe that there is no loss of generality in our notion of initial goal since any arbitrary term t could be used as an initial goal by just adding a new rule, $\mathsf{goal}(x_1, \ldots, x_n) \to t$, where goal is a fresh function symbol with $\mathit{Var}(t) = \{x_1, \ldots, x_n\}$, and then considering $\mathsf{goal}(x_1, \ldots, x_n)$ as initial goal. In the same vein, if one wants to prove termination for more than one initial goal, then this can be encoded by adding several goal rules, or by taking the conjunction of individual goal proofs.

Two key ingredients of the dependency pair approach, as explained in detail in Section 4.1, are the notion of *dependency pair* and the notion of *chain*. In the Goal-directed extension of dependency pairs, the definition of pair remains the same (cf. Definition 4.2). The auxiliary notion of *initial pairs* denotes the pairs that match an initial goal (below, recall the notion of marked term $t^{\#}$ introduced in Definition 4.2).

Definition 5.6 (Initial Pairs)**.** *Let $\mathcal{R}$ be a TRS and t_0 be an initial goal. The associated set of initial pairs from t_0, $\mathcal{I}_{t_0}$, is defined as $\{s \to t \in DP(\mathcal{R}) \mid s \in \lceil t_0 \rceil\}$.*

On the other hand, the notion of chain is restricted to consider only *reachable* chains. In order to formalize our definition of chains, we first introduce the notion of *reachable calls* from a given term. Formally, given a TRS $\mathcal{R}$ and a term t, we define the set of reachable calls, $calls_{\mathcal{R}}(t)$, from t in $\mathcal{R}$ as follows:

$$calls_{\mathcal{R}}(t) = \{\ s|_p \ \mid\ t \to^*_{\mathcal{R}} s, \text{ and } \mathsf{root}(s|_p) \in \mathcal{D} \text{ for some position } p\ \}$$

Also, given a set of terms T, we let $calls_{\mathcal{R}}(T) = \bigcup_{t \in T} calls_{\mathcal{R}}(t)$.

The goal-directed chains of a TRS $\mathcal{R}$ are the reachable subset of the chains of $\mathcal{R}$. This is formalized below.

Definition 5.7 (Goal-directed chain)**.** *Let $\mathcal{R}$ and $\mathcal{P}$ be TRSs over the signatures $\mathcal{F}$ and $\mathcal{F}^{\sharp}$, respectively. Let t_0 be an initial goal. A (possibly infinite) sequence of pairs $s_1 \to t_1$, $s_2 \to t_2$, ... from $\mathcal{P}$ is a $(t_0, \mathcal{P}, \mathcal{R})$-chain if there is a substitution $\sigma : \mathcal{V} \mapsto \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that the following conditions hold:*[1]

1. *there exists a term $s \in calls_{\mathcal{R}}(\lceil t_0 \rceil)$ such that $s^{\sharp} = s_1\sigma$ and*
2. *$t_i\sigma \to^*_{\mathcal{R}} s_{i+1}\sigma$ for every two consecutive pairs in the sequence.*

The chain is minimal iff all $t_i\sigma$ are terminating w.r.t. $\mathcal{R}$.

Note that the only difference with the standard notion of chain (cf. Definition 4.3) is that only chains which are *reachable* from (an instance of) the initial goal are considered. Put differently, we do not demand that goal-directed chains start in an initial goal; our chains can start anywhere, i.e., we consider suffixes of (possibly infinite) chains starting from an initial goal.

The following lemma states this more formally.

Lemma 5.8. *Let $\mathcal{R}$ be a TRS, $\mathcal{P} \subseteq DP(\mathcal{R})$ be a TRS, t_0 be an initial goal, and the sequence $S : s_1 \to t_1, \ldots, s_n \to t_n$ be a possibly infinite $(t_0, \mathcal{P}, \mathcal{R})$-chain. Let $\mathcal{I}_{t_0} \subseteq DP(\mathcal{R})$ be the initial pairs from t_0.*

There is a term $t \in \lceil t_0 \rceil$, constructor substitution σ, and a possibly empty prefix $S' : s'_1 \to t'_1, \ldots, s'_m \to t'_m$ of pairs of $\mathcal{P}$, such that the sequence $S'S$ constitutes a $(t_0, \mathcal{P}, \mathcal{R})$-chain, and $s'_1\sigma = t^{\#}$ for the first pair $s'_1 \to t'_1$ of $S'S$.

[1]As in [Arts and Giesl, 2000], we assume fresh variables in every (occurrence of a) dependency pair and that the domain of substitutions may be infinite.

Proof. By definition of goal-directed chain, there is a term $s \in \mathit{calls}_{\mathcal{R}}(\lceil t_0 \rceil)$ and substitution σ such that $s_1\sigma = s^{\#}$. By definition of *calls*, there is some term $t \in \lceil t_0 \rceil$ such that $t \to^*_{\mathcal{R}} C[s]$.

If $t = s$, then $s_1\sigma = s^{\#} = u^{\#}$. It follows that σ is constructor since both $u^{\#}$ and s_1 are constructor terms except at the root. Trivially, S' is the empty prefix, and we are done.

Otherwise, we must show that there exists a sequence $S' : s'_1 \to t'_1, \ldots, s'_m \to t'_m$ such that $S'S$ is a $(t_0, \mathcal{P}, \mathcal{R})$-chain. We show how to build this sequence from the derivation $t \to^*_{\mathcal{R}} C[s]$. By definition u is of the form $t_0\theta = \mathtt{f}(x_1, \ldots, x_n)\theta$ for some constructor substitution θ. Then this sequence contains a first rewriting step given at the root: $t \overset{\epsilon}{\to}_{l_1 \to r_1} t'$, and there is a position p such that either $t'|_p = s$, or there is a subterm $t'|_p \overset{\epsilon}{\to}_{l_2 \to r_2} t'' \to^*_{\mathcal{R}} C[s]$. From this step we can extract the first pair $s'_1 \to t'_1$, which is $l_1^{\#} \to (r_1|_p)^{\#}$. Moreover, note that in the case where $t'|_p = s$,we have $S' = s'_1 \to t'_1$, and $S'S$ constitutes a chain since 1) the first condition of the definition of goal-directed chain holds by construction, and 2) $t'|_p = s$ implies that $(r_1|_p)^{\#}\rho = t'_1\rho = s_1\sigma$ for some substitution ρ, (since we assume fresh variables in every occurrence of a dependency pair), so the second condition of the definition of goal-directed chain holds too. In the second case we repeat the process from t'', i.e., there must be a position p' such that either $t''|_{p'} = s$ or $t''|_{p'} \overset{\epsilon}{\to}_{l_3 \to r_3} \circ \to^*_{\mathcal{R}} C[s]$. Then we can define $s'_2 \to t'_2$ to be $l_2^{\#} \to (r_2|'_p)^{\#}$, and if $t''|_{p'} \neq s$ continue repeating this process, until eventually we reach s (as guaranteed by the definition of *calls*). □

Our termination criterion states that every derivation starting from an initial goal t_0 is finite if and only if there are no infinite chains.

Theorem 5.9 (Termination Criterion)**.** *Let $\mathcal{R}$ be a TRS and t_0 be an initial goal. All derivations starting from a term in $\lceil t_0 \rceil$ in $\mathcal{R}$ are finite iff there are no infinite minimal $(t_0, DP(\mathcal{R}), \mathcal{R})$-chains.*

Proof. This is a trivial extension of the proof of Theorem 6 in [*Arts and Giesl*, 2000].

The $(\Rightarrow)$ direction requires us to prove that an infinite derivation issuing from $\lceil t_0 \rceil$ can be extracted from an infinite minimal $(t_0, DP(\mathcal{R}), \mathcal{R})$-chain. Given a sequence of pairs S constituting a minimal $(t_0, DP(\mathcal{R}), \mathcal{R})$-chain, by Lemma 5.8 there is a prefix S' such that $S'S$ is a chain starting from an instance of t_0. By the proof in [*Arts and Giesl*, 2000], from $S'S$ we can construct the desired infinite derivation starting from an instance of t_0.

For the $(\Leftarrow)$ direction, building an infinite minimal goal-directed chain from an infinite derivation starting from a term in $\lceil t_0 \rceil$ is required. Here the proof of [*Arts and Giesl*, 2000] suffices without changes, as any infinite term in $\lceil t_0 \rceil$ is minimal by construction, and by the mentioned proof there is an infinite minimal standard chain which satisfies both conditions of Definition 5.7. □

As in the standard DP framework [*Giesl et al.*, 2005c], and in order to ease the automation of the proof search, we introduce a *goal-directed* DP (GDP) framework as follows:

Definition 5.10 (GDP problems and processors)**.** *A GDP problem is a tuple* $(t_0, \mathcal{P}, \mathcal{R}, f)$ *consisting of two TRSs* $\mathcal{R}$ *and* $\mathcal{P}$ *over the signatures* $\mathcal{F}$ *and* $\mathcal{F}^\sharp$*, respectively, an initial goal* t_0 *for* $\mathcal{R}$*, and a minimality flag* $f \in \{\mathbf{m}, \mathbf{a}\}$ *where* $\mathbf{m}$ *and* $\mathbf{a}$ *stand for "minimal" and "arbitrary", respectively. A GDP problem is finite if there is no associated infinite (minimal if* f *is* $\mathbf{m}$*)* $(t_0, \mathcal{P}, \mathcal{R})$*-chain, and infinite if it is not finite or if* $\lceil t_0 \rceil$ *does not terminate in* $\mathcal{R}$*.*

A GDP processor *is a function Proc which takes a GDP problem and returns either a new set of GDP problems or fails. Proc is sound if for any GDP problem* $\mathcal{M}$*,* $\mathcal{M}$ *is finite whenever all GDP problems in Proc*$(\mathcal{M})$ *are finite.*

As usual, we construct a tree whose root is labeled with the problem $(t_0$, $DP(\mathcal{R})$, $\mathcal{R}$, $\mathbf{m})$ and whose nodes are produced by application of sound GDP processors. $\lceil t_0 \rceil$ is terminating in $\mathcal{R}$ if all the leaf nodes of this tree are finite problems.

5.1.1 Dependency Graphs

The notion of dependency graph (cf. Definition 4.5) is adapted to the goal-directed extension simply by replacing the use of DP chains with the goal-directed version of chain.

Definition 5.11 (Goal–directed Dependency Graph)**.** *For a GDP problem* $(t_0, \mathcal{P}, \mathcal{R}, f)$ *its dependency graph is a directed graph where the nodes are the pairs of* $\mathcal{P}$*, and there is an edge from* $s \to t \in \mathcal{P}$ *to* $u \to v \in \mathcal{P}$ *iff* $s \to t, u \to v$ *is a* $(t_0, \mathcal{P}, \mathcal{R})$*-chain.*

Even though we consider the standard definition of dependency graph, note that since our notion of chain is different, the GDP dependency graph usually contains less pairs than the standard dependency graph. Concretely, all the edges which are not reachable from the initial goal are not included in the GDP dependency graph, as we will see in the following.

As mentioned earlier, dependency graphs are not generally computable. Several approximations have been defined in the literature. Instead of adapting one of these approximations, we show how any arbitrary approach can easily be reused in our context.

Theorem 5.12 (Goal–Directed Estimated Dependency Graph)**.** *Let* $(t_0, \mathcal{P}, \mathcal{R}, f)$ *be a GDP problem such that* $\mathcal{P} \subseteq DP(\mathcal{R})$*. Let* $\mathcal{I}_{t_0} \subseteq DP(\mathcal{R})$ *be the set of initial pairs from* t_0*. Let* $\mathcal{G}_0$ *and* $\mathcal{G}$ *be the dependency graphs of the DP problems* $(DP(\mathcal{R}), \mathcal{R}, f)$ *and* $(\mathcal{P}, \mathcal{R}, f)$*, respectively.*

Define $\mathcal{G}_{\mathcal{I}_{t_0}}$ *to be the subgraph of* $\mathcal{G}_0$ *constituted by the nodes which are reachable from a pair in* $\mathcal{I}_{t_0}$*. The dependency graph of the GDP problem* $(t_0, \mathcal{P}, \mathcal{R}, f)$ *is the intersection of* $\mathcal{G}$ *and* $\mathcal{G}_{\mathcal{I}_{t_0}}$*.*

Proof. Since every $(t_0, \mathcal{P}, \mathcal{R})$-chain is also a $(\mathcal{P}, \mathcal{R})$-chain, $\mathcal{G}$ is by construction an overestimation of the $(t_0, \mathcal{P}, \mathcal{R})$ graph.

By definition the nodes of $\mathcal{G}$ are a subset of the nodes in $\mathcal{G}_o$. We only need to show that the edges removed do not constitute $(t_0, \mathcal{P}, \mathcal{R})$-chains, which follows from the first condition in Definition 5.7. □

With the Theorem above, it is easy to adapt any existing estimation of the standard dependency graph (e.g. the one from Definition 4.28) to the GDP setting.

Example 5.13. *Consider the following dependency graph $\mathcal{G}_0$ whose nodes are labeled with $(0), (1), \ldots, (7)$:*

$$\begin{tikzcd}
1 \arrow[r, bend left] & 2 \arrow[l, bend left] & 3 \arrow[r, bend left] & 4 \arrow[l, bend left] & 0 \arrow[l] \arrow[r] & 5 \arrow[r] & 6 \arrow[r, bend left] & 7 \arrow[l, bend left]
\end{tikzcd}$$

Given a GDP problem where the only initial pair is (0), we have that pairs (1) and (2) do not belong to its dependency graph since they are not reachable.

Example 5.14. *Recall the TRS R_{inc} from Example 5.4, and consider the GDP problem $(\mathsf{p_{in}}(X), DP(\mathcal{R}_{inc}), \mathcal{R}_{inc}, f)$, where $DP(\mathcal{R}_{inc})$ contains the following set of dependency pairs:*

$$\mathsf{P_{in}}(\mathsf{g}(X)) \to \mathsf{P_{in}}(X) \tag{5.1}$$
$$\mathsf{P_{in}}(\mathsf{g}(X)) \to \mathsf{U_3}(\mathsf{p_{in}}(X), X) \tag{5.2}$$
$$\mathsf{P_{in}}(X) \to \mathsf{Q_{in}}(\mathsf{f}(Y)) \tag{5.3}$$
$$\mathsf{P_{in}}(X) \to \mathsf{U_1}(\mathsf{q_{in}}(\mathsf{f}(Y)), X) \tag{5.4}$$
$$\mathsf{U_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) \to \mathsf{U_2}(\mathsf{p_{in}}(Y), X, Y) \tag{5.5}$$
$$\mathsf{U_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) \to \mathsf{P_{in}}(Y) \tag{5.6}$$

The initial pairs are (5.1), (5.2), (5.3) and (5.4). We obtain the following estimated dependency graph

$$\begin{tikzcd}
 & (5.6) \arrow[dl, dotted] \arrow[d, dotted] \arrow[dr, dotted] \arrow[dd, dotted, bend right] & (5.5) \\
(5.4) & (5.3) & (5.2) \\
 & (5.1) \arrow[ul] \arrow[u] \arrow[ur] \arrow[loop below] &
\end{tikzcd}$$

where pairs (5.6) and (5.5) are unreachable, since there is no path connecting them with a node corresponding to an initial pair, and therefore do not belong to the goal–directed dependency graph.

By using the estimated dependency graph, it is immediate to define a sound GDP processor that takes a GDP problem $(t_0, \mathcal{P}, \mathcal{R}, f)$ and divides the problem into its strongly connected components (SCC) as usual. The following is the standard definition of the dependency graph processor, the only change is that it uses the goal–directed dependency graph instead of the standard one, and therefore it returns fewer subproblems when some SCCs are not reachable from the initial goal.

Theorem 5.15 (Dependency Graph Processor)**.** *Given a GDP problem $(t_0, \mathcal{P}, \mathcal{R}, \mathcal{P}_0)$ with $\mathcal{P}$ a finite set of pairs, let Proc be the processor that returns the set of problems $\{(t_0, \mathcal{P}_1, \mathcal{R}, \mathcal{P}_0), \dots, (t_0, \mathcal{P}_n, \mathcal{R}, \mathcal{P}_0)\}$, where $\mathcal{P}_1, \dots, \mathcal{P}_n$ are the sets of nodes of every non-trivial strongly connected component (SCC) in an overestimation of the goal–directed dependency graph. Then, Proc is sound.*

Proof. The proof is virtually identical to the proof of the NDP dependency graph processor of Theorem 4.41, since both are straightforward instances of the standard dependency graph processor.

We prove the theorem by contradiction. Assume that `Proc` is not sound. Then there is an GDP problem $\tau = \langle t_0, \mathcal{P}, \mathcal{R}, f \rangle$ such that every subproblem $\tau \in \texttt{Proc}(\tau)$ is finite whereas τ is not. Then there is an infinite chain $\mathcal{P}_1 \subseteq \mathcal{P}$ such that –since $\mathcal{P}$ is finite– a tail $B \subseteq \mathcal{P}_1$ of pairs is repeated infinitely. The nodes corresponding to B form a cycle in the goal–directed dependency graph, and by definition also in the overestimation. This cycle must belong to some SCC; the subproblem $\tau_i \in \texttt{Proc}(\tau)$ corresponding to this SCC is not finite, since B is also an infinite chain of τ_i. Thus we reach a contradiction with the assumption that all the subproblems are finite, which proves the soundness of `Proc`. □

5.1.2 Usable Rules

Another way for removing pairs from $\mathcal{P}$ is based on the notion of *reduction pair* $(\succsim, \succ)$ (cf. Definition 4.6). For this purpose, recall the notion of argument filtering from Section 4.2.4, given in Definition 4.35.

Theorem 5.16 (Reduction Pair Processor)**.** *Let $(\succsim, \succ)$ be a reduction pair and π be an argument filtering. Given a GDP problem $(t_0, \mathcal{P}, \mathcal{R}, f)$, if Proc returns:*

- $(t_0, \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f)$, *if* $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi} = \mathcal{P}$ *and* $\mathcal{R}_{\succsim_\pi} = \mathcal{R}$
- $(t_0, \mathcal{P}, \mathcal{R}, f)$, *otherwise*

then Proc is sound and complete.

Proof. Completeness follows from the fact that *Proc* does not introduce any new rules or pairs. Soundness is already shown in [*Giesl et al.*, 2005c] for standard minimal $(\mathcal{P}, \mathcal{R})$-chains. Since every $(t_0, \mathcal{P}, \mathcal{R})$-chain is also a $(\mathcal{P}, \mathcal{R})$-chain, this processor is sound in the GDP framework too. □

Roughly speaking, this processor can be used to remove the strictly decreasing pairs of $\mathcal{P}$ when the remaining pairs of $\mathcal{P}$ and all rules of $\mathcal{R}$ are weakly decreasing. In fact, weak decreasingness is not required for all the rules but only for the *usable* rules [*Giesl et al.*, 2006b]. The usable rules are those rules which may be needed to connect dependency pairs in a chain. For GDP problems a notion of usable rules can be defined which does not include rules that are not reachable from the initial goal.

There are several approaches for approximating the usable rules of a problem. In this section we show how these can be adapted to our goal-directed setting.

In the following, given a graph $\mathcal{G}$ and two sets of nodes $\mathcal{I}$ and $\mathcal{P}$, $PATH_{\mathcal{G}}(\mathcal{I},\mathcal{P})$ denotes the smallest set of nodes of $\mathcal{G}$ which contains both $\mathcal{I}$ and $\mathcal{P}$ and all the nodes which are in a path from some node in $\mathcal{I}$ to some node in $\mathcal{P}$ in $\mathcal{G}$.

Definition 5.17 (Estimated Goal-Directed Usable Rules w.r.t. an Argument Filtering)**.** *Let $(t_0, \mathcal{P}, \mathcal{R}, f)$ be a GDP problem, $\mathcal{P}_0$ its initial pairs, π an argument filtering, and $\mathcal{G}_0$ a (standard) estimated dependency graph for the DP problem $(DP(\mathcal{R}), \mathcal{R}, f)$. A function $\mathcal{U}^{\pi}_{\mathcal{R}}$ estimates the usable rules of a TRS $\mathcal{R}$ for a term t w.r.t. an argument filtering π, written $\mathcal{U}^{\pi}_{\mathcal{R}}(t)$ if, for any constructor substitution σ and term u:*

$$\textit{If } t\sigma \to^*_{\mathcal{R}} u \textit{ then } \pi(t\sigma) \to^*_{\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(t))} \pi(u) \textit{ and } \mathcal{U}^{\pi}_{\mathcal{R}}(u) \subseteq \mathcal{U}^{\pi}_{\mathcal{R}}(t) \tag{5.7}$$

We lift $\mathcal{U}^{\pi}_{\mathcal{R}}$ to sets of rules in the usual manner: for any such set $\mathcal{Q}$, $\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{Q}) = \bigcup_{l\to r\in\mathcal{Q}} \mathcal{U}^{\pi}_{\mathcal{R}}(r)$.

The goal-directed usable rules of $\mathcal{P}$ in $\mathcal{R}$ w.r.t. π and t_0, $\mathcal{GU}(t_0, \mathcal{P}, \mathcal{R}, \pi)$, are defined as:

- *$\mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P}_0)$, if $Var(\pi(t)) \subseteq Var(\pi(s))$ for all $s \to t \in PATH_{\mathcal{G}_0}(\mathcal{P}_0, \mathcal{P})$,*
- *$\mathcal{R}$, otherwise.*

The definition of goal-directed usable rules above is parameterized with the regular estimation of usable rules. Any such estimation can be used as long as it satisfies condition (5.7) in page 85. This condition is pretty reasonable and we argue that any standard estimation of the usable rules in the literature can be shown to satisfy it. To be concrete, we show now that the estimation of [Giesl *et al.*, 2006b], introduced in Chapter 4, Definition 4.45, does indeed satisfy it.

Lemma 5.18. *Let $\mathcal{R}$ be a TRS, π an argument filtering, and σ a constructor substitution. Let $\mathcal{U}^{\pi}_{\mathcal{R}}(t)$ be the estimation of the usable rules for term t in $\mathcal{R}$ w.r.t. π defined as in Definition 4.45. For all terms t, u, we have the following:*

1. *If $t\sigma \to_{\mathcal{R}} u$, then either $\pi(t) = \pi(u)$, or $\pi(t) \to_{\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(t))} \pi(u)$ and $\mathcal{U}^{\pi}_{\mathcal{R}}(u) \subseteq \mathcal{U}^{\pi}_{\mathcal{R}}(t)$,*
2. *If $t\sigma \to^*_{\mathcal{R}} u$, then $\pi(t) \to^*_{\pi(\mathcal{U}^{\pi}_{\mathcal{R}}(t))} \pi(u)$ and $\mathcal{U}^{\pi}_{\mathcal{R}}(u) \subseteq \mathcal{U}^{\pi}_{\mathcal{R}}(t)$.*

Proof. This lemma is a slightly modified version of Lemma 23 in [Giesl *et al.*, 2006b]. The proof is mostly routine. In the following, let $\mathcal{U}^{\pi}_{\mathcal{R}}(\sigma)$ denote $\cup_x \mathcal{U}^{\pi}_{\mathcal{R}}(\sigma(x))$. Also, we use the following fact, which holds for any t (a proof can be obtained by straightforward structural induction on t):

$$\mathcal{U}^{\pi}_{\mathcal{R}}(t\sigma) = \mathcal{U}^{\pi}_{\mathcal{R}}(t) \text{ if } \sigma \text{ is a constructor substitution.} \tag{5.8}$$

1. By induction on the position of the reduction, which must be in t since σ is constructor. Hence t is of the form $\mathtt{f}(t_1, \ldots, t_n)$. For the base case, assume that the reduction is in the root position, and we have $t\sigma = l\sigma' \to_{l\to r} r\sigma' = v$ for some rule $l \to r \in \mathsf{Def}_{\mathcal{R}}(\mathsf{f}) \subseteq \mathcal{U}^{\pi}_{\mathcal{R}}(t)$. Thus, $\pi(t\sigma) \to_{\mathcal{U}^{\pi}_{\mathcal{R}}(t)} \pi(r\sigma')$. Let

$u = \pi(r\sigma')$. Now to show the second part of the claim, we observe that every term u in the range of σ' is a subterm of $t\sigma$, and hence $\mathcal{U}^\pi_\mathcal{R}(u) = \mathcal{U}^\pi_\mathcal{R}(r\sigma') \subseteq \mathcal{U}^\pi_\mathcal{R}(r) \cup \mathcal{U}^\pi_\mathcal{R}(t\sigma)$. By fact (5.8), we can remove the constructor σ in the second term, obtaining $\mathcal{U}^\pi_\mathcal{R}(u) \subseteq \mathcal{U}^\pi_\mathcal{R}(r) \cup \mathcal{U}^\pi_\mathcal{R}(t)$. Expanding the definition of the first term, we have $\mathcal{U}^\pi_\mathcal{R}(r) \subseteq \mathsf{Def}_\mathcal{R}(\mathsf{f}) \cup \bigcup_{l' \to r' \in \mathsf{Def}_\mathcal{R}(\mathsf{f})} \subseteq \mathcal{U}^\pi_\mathcal{R}(t)$, and hence we conclude $\mathcal{U}^\pi_\mathcal{R}(u) \subseteq \mathcal{U}^\pi_\mathcal{R}(t)$.

For the induction case, we have

$$t\sigma = \mathtt{f}(t_1\sigma \dots t_i\sigma \dots t_n\sigma) \to_\mathcal{R} \mathtt{f}(t_1\sigma \dots u_i\sigma \dots t_n\sigma) = u$$

If $\pi(t\sigma) = \pi(u)$ then we are done, otherwise either $i \in \pi(\mathtt{f})$ or $\pi(\mathtt{f}) = i$. By the induction hypothesis, $\pi(t_i\sigma) \to_{\pi(\mathcal{U}^\pi_\mathcal{R}(t))} \pi(u_i)$, and it follows by monotonicity of $\to$ that $\pi(t\sigma) \to_{\pi(\mathcal{U}^\pi_\mathcal{R}(t))} \pi(u)$. For the second part of the claim, we have:

- $\mathcal{U}^\pi_\mathcal{R}(u_i) \subseteq \mathcal{U}^\pi_\mathcal{R}(t_i)$ (by the induction hypothesis)
- $\mathcal{U}^\pi_\mathcal{R}(t_i) \subseteq \mathcal{U}^\pi_\mathcal{R}(t)$ (by definition of $\pi(\mathcal{U}^\pi_\mathcal{R}(t_i)))$

It is easy to see that $\mathcal{U}^\pi_\mathcal{R}(u) \subseteq \mathcal{U}^\pi_\mathcal{R}(t)$.

2. Trivial by induction on the length of the derivation using the previous result. □

Example 5.19. *Let us consider again the dependency graph $\mathcal{G}_0$ from Example 5.13, together with the set $\mathcal{I} = \{(0)\}$. Then,*

- *if $\mathcal{P}$ is the SCC $\{(6),(7)\}$ then $PATH_{\mathcal{G}_0}(\mathcal{I},\mathcal{P}) = \{(0),(5),(6),(7)\}$;*
- *if $\mathcal{P} = \{(3),(4)\}$, then $PATH_{\mathcal{G}_0}(\mathcal{I},\mathcal{P}) = \{(0),(3),(4)\}$.*

The following lemma states that for all $(t_0,\mathcal{P},\mathcal{R})$-chain, given a argument filtering π, there is a $(\pi(t_0),\pi(\mathcal{P}),\pi(\mathcal{R}))$-chain where the only $\pi(\mathcal{R})$ rules intervening are the usable rules. This justifies the use of goal–directed usable rules in termination proofs.

Lemma 5.20. *Let t_0 be an initial goal and π an argument filtering. Let $s_1 \to t_1, s_2 \to t_2, \dots$ be a $(t_0,\mathcal{P},\mathcal{R})$ chain. If every $s_i \to t_i$ belongs to $DP(\mathcal{R})$, then $\pi(s_1) \to \pi(t_1), \pi(s_2) \to \pi(t_2), \dots$ is a $(\pi(t_0),\pi(\mathcal{P}),\pi(\mathcal{GU}(t_0,\mathcal{P},\mathcal{R},\pi))))$-chain.*

Proof. We focus on the extended chain $\mathcal{P}'\mathcal{P}$ starting from an instance $\lceil t_0 \rceil$ of the initial term, as described in Lemma 5.8. By said lemma, there is an extended chain $\mathcal{P}'\mathcal{P} = l_0 \to r_0, l_1 \to r_1 \dots$ with $l_0 \to r_0$ an initial pair, and a constructor instance t of the initial goal, $t \in \lceil t_0 \rceil$, such that $t^\# = l_0\sigma$. We can construct a derivation of the form:

$$t^\# \xrightarrow{\epsilon}_{l_0 \to r_0} u_0 \xrightarrow{>\epsilon*}_\mathcal{R} t_1 \xrightarrow{\epsilon}_{l_1 \to r_1} u_1 \xrightarrow{>\epsilon*}_\mathcal{R} \cdots$$

In order to prove the lemma, we must show that there is a substitution σ such that for all $\pi(t_i)$, $\pi(t_i) \to^*_{\pi(\mathcal{GU}(t_0,\mathcal{P},\mathcal{R},\pi))} \pi(s_{i+1})$. In such case, we can construct a filtered version of the derivation above:

$$\pi(t^\#) \xrightarrow{\epsilon}_{\pi(l_0) \to \pi(r_0)} \pi(u_0) \xrightarrow{>\epsilon*}_{\pi(\mathcal{GU})} \pi(t_1) \xrightarrow{\epsilon}_{\pi(l_1) \to \pi(r_1)} \pi(u_1) \xrightarrow{>\epsilon*}_{\pi(\mathcal{GU})} \cdots$$

where we abbreviate $\mathcal{GU}(t_0, \mathcal{P}, \mathcal{R}, \pi)$ by $\mathcal{GU}$.

If $\pi(\mathcal{P})$ contains extra variables, then $\mathcal{GU} = \mathcal{R}$ and the result follows trivially from Lemma 4.38.

Otherwise, we assume that $\pi(\mathcal{P})$ contains no extra variables and perform induction on the index i of the dependency pair $l_i \to r_i$ to prove that $\pi(u_i) \xrightarrow{>\epsilon *}_{\pi(\mathcal{GU})} \pi(t_{i+1})$ and $\mathcal{U}^{\pi}_{\mathcal{R}}(t_{i+1}) \subseteq \mathcal{GU}$.

For the base case we consider the first dependency pair, and the segment $\pi(t^{\#}) \xrightarrow{\epsilon}_{\pi(l_0) \to \pi(r_0)} \pi(u_0) \xrightarrow{>\epsilon *}_{\pi(\mathcal{GU})} \pi(t_1)$ of the derivation. From the definitions of initial goal and the $\lceil\ \rceil$ operator, t is of the form $\mathtt{f}(x_1 \ldots x_n)\theta$ for some defined symbol $\mathtt{f}$ and constructor substitution θ. As $t^{\#} = l_0\sigma$, σ is a constructor substitution. We have $\pi(u_0) = \pi(r_0\sigma)$. As σ is a constructor substitution, and $l_0 \to r_0$ an initial pair, $\mathcal{U}^{\pi}_{\mathcal{R}}(t) \in \mathcal{GU}$ and condition (5.7) in page 85 applies. We have $\pi(u_0) \to^*_{\pi(\mathcal{GU})} \pi(t_1)$ and $\mathcal{U}^{\pi}_{\mathcal{R}}(t_1) \subseteq \mathcal{U}^{\pi}_{\mathcal{R}}(r_0) \subseteq \mathcal{GU}$.

For the inductive case consider the segment

$$t^{\#}\ (\xrightarrow{\epsilon}_{\pi(\mathcal{P}'\mathcal{P})} \cdot \xrightarrow{>\epsilon}_{\pi(\mathcal{GU})})^*\ \pi(t_i) \xrightarrow{\epsilon}_{\pi(l_i) \to \pi(r_i)} \pi(u_i) \xrightarrow{>\epsilon *}_{\pi(\mathcal{GU})} \pi(t_{i+1})$$

We have $u_i = r_i\theta$ for some substitution θ. By definition, $\mathcal{U}^{\pi}_{\mathcal{R}}(r_i) \subseteq \mathcal{GU}$. By the induction hypothesis, $\mathcal{U}^{\pi}_{\mathcal{R}}(t_i) \subseteq \mathcal{GU}$, hence $\mathcal{U}^{\pi}_{\mathcal{R}}(v_i) \subseteq \mathcal{GU}$ for every subterm v_i of t_i in the range of θ. As $l_i \to r_i$ contains no extra variables, it follows that $\mathcal{U}^{\pi}_{\mathcal{R}}(u_i) \subseteq \mathcal{GU}$. Applying condition (5.7) with $t = u_i$ and σ the empty substitution, proves that $\pi(u_i) \xrightarrow{>\epsilon *}_{\pi(\mathcal{GU})} \pi(t_{i+1})$ and $\mathcal{U}^{\pi}_{\mathcal{R}}(t_i) \subseteq \mathcal{GU}$, and we are done. □

The goal-directed usable rules coincide with the usable rules for all the chains from an initial pair to the pairs in $\mathcal{P}$. While this means that they are a superset of the usable rules of $\mathcal{P}$, they are still advantageous as they are applicable in cases where the usable rules are not. In particular, *minimality* is not required. This is critical in Section 5.3 where we consider the termination of narrowing as a problem of relative termination, since in this context minimality does not generally hold.

The reduction processor with usable rules in the Theorem below makes use of the goal-directed usable rules when minimality is not ensured. Otherwise, the standard usable rules are employed.

Theorem 5.21 (Reduction Pair Processor with Goal-Directed Usable Rules w.r.t. an Argument Filtering)**.** *Let $(\succsim, \succ)$ be a reduction pair and π be an argument filtering. Given a GDP problem $(t_0, \mathcal{P}, \mathcal{R}, f)$ such that $\mathcal{P} \subseteq DP(\mathcal{R})$, then Proc returns:*

- $(t_0, \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, \mathbf{m})$*, if f is* $\mathbf{m}$*, $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi} = \mathcal{P}$, $\mathcal{R}_{\succsim_\pi} \supseteq \mathcal{U}^{\pi}_{\mathcal{R}}(\mathcal{P})$, and $\succsim$ is $\mathcal{C}_{\mathcal{E}}$-compatible;*
- $(t_0, \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R}, f)$*, if $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi} = \mathcal{P}$, and $\mathcal{R}_{\succsim_\pi} \supseteq \mathcal{GU}(t_0, \mathcal{P}, \mathcal{R}, \pi)$;*
- $(t_0, \mathcal{P}, \mathcal{R}, f)$*, otherwise;*

Proc is sound and complete.

Proof. The processor is complete by definition since it adds no new rules or pairs. The soundness of the first case is already proven in [*Giesl et al.*, 2006b] for a standard DP framework, and can be generalized to the GDP framework since every $(t_0, \mathcal{P}, \mathcal{R})$-chain is also a $(\mathcal{P}, \mathcal{R})$-chain. We give the proof of soundness for the novel second case.

Let us suppose that the original problem is not finite and there is an infinite $(t_0, \mathcal{R}, \mathcal{P})$-chain $s_1 \to t_1, s_2 \to t_2, \ldots$. By Lemma 5.20 we know that $\pi(s_1) \to \pi(t_1), \pi(s_2) \to \pi(t_2), \ldots$ is also a $(\pi(t_0), \pi(\mathcal{P}), \pi(\mathcal{GU}(t_0, \mathcal{P}, \mathcal{R}, \pi)))$-chain. Since $\mathcal{P} = \mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi}$, pairs of $\mathcal{P}_{\succ_\pi}$ cannot occur infinitely since they are decreasing, otherwise the chain cannot be infinite. Hence they can be safely removed as there is an infinite suffix of the infinite chain in which they do not occur. □

Example 5.22. *Consider the following GDP problem*

$$(\mathsf{goal}(x), \{\mathsf{ADD}(\mathsf{s}(x), y) \to \mathsf{ADD}(x, y)\}, \mathcal{R}_{add}, \mathbf{a})$$

where $\mathcal{R}_{add}$ is the following set of rules:

$$\begin{aligned} \mathsf{goal}(x) &\to \mathsf{add}(x, \mathsf{gen}) \\ \mathsf{add}(\mathsf{s}(x), y) &\to \mathsf{s}(\mathsf{add}(x, y)) \\ \mathsf{add}(\mathsf{zero}, y) &\to y \\ \mathsf{gen} &\to \mathsf{s}(\mathsf{gen}) \\ \mathsf{gen} &\to \mathsf{zero} \end{aligned}$$

and as the reader can check, $DP(\mathcal{R}_{add})$ is the set of pairs:

$$\begin{aligned} \mathsf{GOAL}(x) &\to \mathsf{ADD}(x, \mathsf{gen}) \\ \mathsf{GOAL}(x) &\to \mathsf{GEN} \\ \mathsf{ADD}(\mathsf{s}(x), y) &\to \mathsf{ADD}(x, y) \\ \mathsf{GEN} &\to \mathsf{GEN} \end{aligned}$$

Using the default argument filtering which filters nothing, we have that

- *the usable rules w.r.t.* $\mathsf{goal}(x)$ *include only the rules for* gen.

Using the argument filtering defined as $\pi(\mathsf{add}) = \{1\}$ (i.e., $\pi(\mathsf{add}(x, y)) = \mathsf{add}(x)$ and the identity otherwise), we have that

- *the usable rules w.r.t.* $\mathsf{goal}(x)$ *are the empty set.*

Since the rules of gen *are increasing, the finiteness of the GDP problem can only be proved in the second case.*

As this section has shown, using this notion of usable rules it is straightforward to adapt an existing reduction pair processor to the goal-directed setting. Adapting other processors to the framework is straightforward too, since every GDP chain is a DP chain and, thus, soundness is preserved as long as the processor does not introduce new pairs in the graph.

5.2 Relative Termination

In this section we show that it is possible to cast a relative termination problem as a standard DP problem as long as the systems involved satisfy the condition that they form hierarchical combinations.

Definition 5.23 (Hierarchical Combination [*Ohlebusch*, 2002])**.** *A system $\mathcal{R}_0 \cup \mathcal{R}_1$ is the* hierarchical combination *(HC) of a base $\mathcal{R}_0$ over $\mathcal{F}_0 = \mathcal{D}_0 \uplus \mathcal{C}_0$ and an extension $\mathcal{R}_1$ over $\mathcal{F}_1 = \mathcal{D}_1 \uplus \mathcal{C}_0$ if and only if $\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$ and $\mathcal{C}_0 \cap \mathcal{D}_1 = \varnothing$.*

And the notion of relative termination:

Definition 5.24 (Relative Termination)**.** *Given two relations $\rightarrow_R$ and $\rightarrow_E$ we define the compound relation $\rightarrow_R/\rightarrow_E$ as $\rightarrow_E^* \cdot \rightarrow_R \cdot \rightarrow_E^*$.*

Given two TRSs $\mathcal{R}_1$ and $\mathcal{R}_0$, we say that $\mathcal{R}_1$ terminates w.r.t. $\mathcal{R}_0$ if the relation $\rightarrow_{\mathcal{R}_1} / \rightarrow_{\mathcal{R}_0}$ is terminating, i.e., if every (possibly infinite) $\rightarrow_{\mathcal{R}_0} \cup \rightarrow_{\mathcal{R}_1}$ derivation contains only finitely many $\rightarrow_{\mathcal{R}_0}$ steps.

Note that sequences of $\rightarrow_{\mathcal{R}_0}$ steps are "collapsed" and seen as a single $\rightarrow_{\mathcal{R}_1}/\rightarrow_{\mathcal{R}_0}$ step. Hence, an infinite $\rightarrow_{\mathcal{R}_1} / \rightarrow_{\mathcal{R}_0}$ derivation must contain an infinite number of $\rightarrow_{\mathcal{R}_1}$ steps, and thus by assumption only finite $\rightarrow_{\mathcal{R}_0}$ subderivations.

We say that a term t is $\rightarrow_{\mathcal{R}}$-terminating w.r.t. $\mathcal{B}$ or $(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-terminating, if there is no infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation issuing from t. Similarly, we say that T is $\rightarrow_{\mathcal{R}}$-terminating w.r.t. $\mathcal{B}$ if every term in T is $\rightarrow_{\mathcal{R}}$-terminating w.r.t. $\mathcal{B}$.

We make use of the standard notion of minimal (*non-terminating*) term, i.e., a term which starts an infinite derivation while all its proper subterms are terminating. In our setting, we say that a term that is not $\rightarrow_{\mathcal{R}}$-terminating w.r.t. $\mathcal{B}$ is *$(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-minimal* if all its proper subterms are $\rightarrow_{\mathcal{R}}$-terminating w.r.t. $\mathcal{B}$.

Lemma 5.25. *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs over $\mathcal{F}_{\mathcal{R}}$ and $\mathcal{F}_{\mathcal{B}}$ respectively, such that $\mathcal{R} \cup \mathcal{B}$ is the HC of the base $\mathcal{B}$ and the extension $\mathcal{R}$. Every $(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-minimal term $t_0 \in \mathcal{T}(\mathcal{F}_{\mathcal{B}} \cup \mathcal{F}_{\mathcal{R}}, \mathcal{V})$ starting an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation is of the form $t_0 = \mathsf{f}(u_0 \ldots u_n)$, where f is an n-ary defined symbol from $\mathcal{F}_{\mathcal{R}}$.*

In the following we use $\overline{u}$ as a shorthand for a list of terms $u_1, u_2, \ldots$.

Proof. This follows by a standard minimality argument, showing that every step which introduces a new minimal term is given with a rule of $\mathcal{B}$, and therefore always rooted by a $\mathcal{B}$ symbol. By minimality it follows that these minimal terms always contain a finite number of $\mathcal{R}$ steps, and eventually this becomes a pure $\mathcal{B}$ derivation, contradicting the assumption that this is an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation. A more detailed proof follows.

We proceed by contradiction. Suppose that there is a $(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-minimal term of the form $\mathtt{f}(\overline{u})$ with $\mathsf{f} \in \mathcal{F}_{\mathcal{B}}$. By assumption there are no infinite derivations starting from the subterms $\overline{u}$. Therefore we can assume that there are only a finite number of $\mathcal{R}$ steps induced by these subterms.

W.l.o.g. we assume that the arguments $\overline{u}$ are reduced in one or more steps, including at least one $\rightarrow_{\mathcal{R}}$ step, otherwise this would be a pure $\rightarrow_{\mathcal{B}}$ derivation instead of a $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation. Eventually a reduction step must be given at the root with some rule $\mathsf{f}(\overline{w}) \rightarrow r_1 \in \mathcal{B}$. Now the infinite derivation continues from $r_1\sigma_1$, so there must be a minimal subterm of $r_1\sigma_1$ which starts an infinite derivation. By assumption the subterms $\overline{w}\sigma_1$ are all $(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-terminating. Hence r is of the form $C[t_1]$ such that $t_1\sigma_1$ starts an infinite derivation. Because of the HC condition we have that 1) $\mathsf{root}(t_1) \in \mathcal{F}_{\mathcal{B}}$, and 2) r_1 contains no $\mathcal{R}$ calls. The infinite derivation starting from $t_1\sigma_1$ contains strictly less $\mathcal{R}$ reduction steps than the “parent” infinite derivation starting from t_0.

This process can be repeated infinitely, obtaining always a new minimal non-terminating term rooted by a $\mathcal{B}$ symbol. As the number of $\mathcal{R}$ reduction steps in t_0 is finite, and every t_i contains strictly less $\mathcal{R}$ steps, eventually this becomes a pure $\mathcal{B}$ derivation, which contradicts the assumption. $\square$

Now we state the main result of this section. In order to prove relative termination of a TRS $\mathcal{R}$ w.r.t. a TRS $\mathcal{B}$, as long as they form a hierarchical combination, one only needs to prove that the pairs of $\mathcal{R}$ are strongly decreasing, while the pairs of $\mathcal{B}$ can be ignored, even if $\mathcal{B}$ is not terminating.

Theorem 5.26 (Relative Termination Criterion)**.** *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs such that $\mathcal{R} \cup \mathcal{B}$ is the HC of the base $\mathcal{B}$ and the extension $\mathcal{R}$. Then, $\mathcal{R}$ terminates w.r.t. $\mathcal{B}$ if and only if there are no infinite $(DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$-chains.*

Proof. The proof follows the standard dependency pair proof scheme. First we will show that the criterion is *sufficient*, i.e., we show that for any infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation we can construct an infinite chain.

Let t_0 be the term that starts the infinite reduction. W.l.o.g. let us consider a $(\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}})$-minimal term $t_0 = \mathsf{f}_0(\overline{u_0})$ such that none of its proper subterms starts an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation and by Lemma 5.25 $\mathsf{f}_0 \in \mathcal{R}$. Consider an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ reduction starting from such a term t_0. First the arguments $\overline{u_0}$ are reduced in zero or more $\mathcal{R} \cup \mathcal{B}$ steps to $\overline{v_0}$, and eventually a rewrite rule $\mathsf{f}_0(\overline{w_0}) \rightarrow r_0$ from $\mathcal{R}$ is applied at the root position, i.e., there exists a substitution σ_0 such that $\mathsf{f}_0(\overline{v_0}) = \mathsf{f}_0(\overline{w_0})\sigma_0 \rightarrow r_0\sigma_0$.

Since the subterms coming from $\overline{v_0}$ are by assumption terminating, r_0 must be of the form $C[\mathsf{f}_1(\overline{u_1})]$, and $f_1(\overline{u_1})\sigma_0$ is a $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$-minimal subterm starting an infinite derivation. Again by Lemma 5.25 $\mathsf{f}_1 \in \mathcal{R}$.

The first dependency pair of the chain that we construct is $\mathsf{f}_0^{\#}(\overline{w_0}) \rightarrow \mathsf{f}_1^{\#}(\overline{u_1})$. The rest of the pairs are constructed by a similar reasoning, obtaining a sequence of pairs

$$\mathsf{f}_0^{\#}(\overline{w_0}) \rightarrow \mathsf{f}_1^{\#}(\overline{u_1}), \quad \mathsf{f}_1^{\#}(\overline{w_1}) \rightarrow \mathsf{f}_2^{\#}(\overline{u_2}), \cdots$$

all coming from $DP(\mathcal{R})$. It should be obvious that this sequence is really a $(DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$ chain.

Now we show that the criterion is also *necessary* for relative termination. We prove that any infinite $(DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$ chain corresponds to an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation. Suppose we have a chain of the form:

$$\mathsf{f}_0^{\#}(\overline{s_0}) \rightarrow \mathsf{f}_1^{\#}(\overline{t_1}), \quad \mathsf{f}_1^{\#}(\overline{s_1}) \rightarrow \mathsf{f}_2^{\#}(\overline{t_2}), \quad \mathsf{f}_3^{\#}(\overline{s_3}) \rightarrow \mathsf{f}_3^{\#}(\overline{t_3}), \cdots$$

By definition of chain, there is a substitution σ such that $\mathsf{f}_1^{\#}(\overline{t_1})\sigma \rightarrow^*_{\mathcal{R}\cup\mathcal{B}} \mathsf{f}_1^{\#}(\overline{s_1})$, $\mathsf{f}_2^{\#}(\overline{t_2})\sigma \rightarrow^*_{\mathcal{R}\cup\mathcal{B}} \mathsf{f}_2^{\#}(\overline{s_2})$, and so on.

As every dependency pair $\mathsf{f}_0^{\#}(\overline{s_0}) \rightarrow \mathsf{f}_1^{\#}(\overline{t_1})$ corresponds to a rule $\mathsf{f}_0(\overline{s_0}) \rightarrow C_1[\mathsf{f}_1(\overline{t_1})]$ in $\mathcal{R}$ we can construct the derivation

$$\mathsf{f}_0(\overline{s_0})\sigma \rightarrow_{\mathcal{R}} C_1[\mathsf{f}_1(\overline{t_1})]\sigma \rightarrow^*_{\mathcal{R}\cup\mathcal{B}} C_1[\mathsf{f}_1(\overline{s_1})]\sigma \rightarrow_{\mathcal{R}} C_1[C_2[\mathsf{f}_2(\overline{t_2})]]\sigma \rightarrow^*_{\mathcal{R}\cup\mathcal{B}} \cdots$$

which is an infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation. This concludes the proof. □

The following example shows that the restriction to hierarchical combinations is necessary. Observe that the systems in the example do not form a hierarchical combination.

Example 5.27. *Let $\mathcal{R} = \{\mathsf{f} \rightarrow \mathsf{gen}\}$ and $\mathcal{B} = \{\mathsf{gen} \rightarrow \mathsf{f}\}$ be TRSs, which are trivially terminating. Moreover, the DP Problem $(DP(\mathcal{R}), \mathcal{R}\cup\mathcal{B}) = (\varnothing, \mathcal{R}\cup\mathcal{B})$ is trivially finite (here we assume that* gen *is a constructor symbol in $\mathcal{R}$, hence the set of dependency pairs $DP(\mathcal{R})$ is empty). However, $\mathcal{R}$ is not terminating w.r.t. $\mathcal{B}$ since we have the following infinite derivation:* $\mathsf{f} \rightarrow \mathsf{gen} \rightarrow \mathsf{f} \rightarrow \ldots$

Note also that it does not suffice to prove the absence of *minimal* chains, as the following example suggests:

Example 5.28. *Let $\mathcal{R} = \{\mathsf{f}(\mathsf{s}(x)) \rightarrow \mathsf{f}(x)\}$ and $\mathcal{B} = \{\mathsf{gen} \rightarrow \mathsf{s}(\mathsf{gen})\}$. We have that $DP(\mathcal{R})$ contains a single pair $\mathsf{F}(\mathsf{s}(x)) \rightarrow \mathsf{F}(x)\}$, and there are no infinite minimal chains. However there is an infinite chain with a substitution $\sigma = \{x \mapsto \mathsf{gen}\}$.*

The relative termination criterion can be combined with Theorem 5.9 for relative termination from an initial goal.

Corollary 5.29 (Goal-Directed Relative Termination Criterion)**.** *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs such that $\mathcal{R}\cup\mathcal{B}$ is the hierarchical combination of the base $\mathcal{B}$ and the extension $\mathcal{R}$, and let t_0 be an initial goal. For all $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation $\mathcal{D}$ starting from a term in $\lceil t_0 \rceil$, $\mathcal{D}$ is terminating if and only if there are no infinite $(t_0, DP(\mathcal{R}), \mathcal{R}\cup\mathcal{B})$-chains.*

5.2.1 Beyond hierarchical combinations

In this section we extend the relative termination criterion of the previous sections to consider combinations which are *almost* hierarchical, but not quite. The motivation for this refinement will be clear in the following section, when we consider termination of narrowing.

Consider the TRS $\mathcal{R}$ given by the following rules:

$$\begin{aligned} \mathtt{f}(x,x) &\rightarrow \mathtt{f}(\mathsf{g}(0),x) \\ \mathsf{g}(1) &\rightarrow 1 \end{aligned}$$

together with the TRS $\mathcal{R}_{\mathsf{gen}}$ defined by the rules:

$$\begin{aligned} \mathsf{gen} &\rightarrow 0 \\ \mathsf{gen} &\rightarrow 1 \\ \mathsf{gen} &\rightarrow \mathsf{g}(0) \end{aligned}$$

The system $\mathcal{R} \cup \mathcal{R}_{\mathsf{gen}}$ does not constitute a hierarchical combination because $\mathcal{R}_{\mathsf{gen}}$ is not a suitable base system, as it contains a call to g, a defined symbol in $\mathcal{R}$. However, observe that this call is actually a *rigid* head normal form, in the sense that any instantiation is still a head normal form.

Definition 5.30 (Rigid Head Normal Form)**.** *A term s is a rigid head normal form with regard to a TRS $\mathcal{R}$ if there is no substitution σ and terms t, t' such that $s\sigma \xrightarrow{>\epsilon *}_{\mathcal{R}} t' \xrightarrow{\epsilon}_{\mathcal{R}} t$.*

It is in general undecidable whether a term t is a rigid head normal form, but it can be overapproximated by means of the *cap* function of Definition 4.28.

Calls to defined symbols of the extension which constitute rigid head normal forms cannot lead to undesired behaviours as in Example 5.27. On the basis of this, we define a class of modular combinations which relaxes the notion of hierarchical combination to allow for these calls, and then prove that the termination criterion of the previous section can be generalized to this class.

Definition 5.31 (Relaxed Hierarchical Combination)**.** *A system $\mathcal{R}_0 \cup \mathcal{R}_1$ is the* relaxed hierarchical combination *(RHC) of a base $\mathcal{R}_0$ over $\mathcal{F}_0 = \mathcal{D}_0 \uplus \mathcal{C}_0$ and an extension $\mathcal{R}_1$ over $\mathcal{F}_1 = \mathcal{D}_1 \uplus \mathcal{C}_0$ if and only if*

- *$\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$ and*
- *for all rule $l \to r \in \mathcal{R}_0$, if $\mathsf{root}(r|_p) \in \mathcal{D}_1$ then $r|_p$ is a rigid head normal form w.r.t. $\mathcal{R}_0 \cup \mathcal{R}_1$.*

Lemma 5.32. *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs over $\mathcal{F}_{\mathcal{R}}$ and $\mathcal{F}_{\mathcal{B}}$ respectively, such that $\mathcal{R} \cup \mathcal{B}$ is the RHC of the base $\mathcal{B}$ and the extension $\mathcal{R}$. Every $(\to_{\mathcal{R}} / \to_{\mathcal{B}})$-minimal term $t_0 \in \mathcal{T}(\mathcal{F}_{\mathcal{B}} \cup \mathcal{F}_{\mathcal{R}}, \mathcal{V})$ starting an infinite $\to_{\mathcal{R}} / \to_{\mathcal{B}}$ derivation is of the form $t_0 = \mathsf{f}(u_0 \dots u_n)$, where f is an n-ary defined symbol from $\mathcal{F}_{\mathcal{R}}$.*

Proof. The proof is an straightforward adaptation of the proof for Lemma 5.25. The defining property which still holds for RHCs is that the right hand sides of $\mathcal{R}_0$ contribute no $\mathcal{R}_1$ redexes, which is guaranteed in this case by the second condition in Definition 5.31.

We proceed by contradiction. Suppose that there is a $(\to_{\mathcal{R}} / \to_{\mathcal{B}})$-minimal term of the form $\mathtt{f}(\overline{u})$ with $\mathsf{f} \in \mathcal{F}_{\mathcal{B}}$. By assumption there are no infinite derivations starting from the subterms $\overline{u}$. Therefore we can assume that there are only a finite number of $\mathcal{R}$ steps induced by these subterms.

W.l.o.g. we assume that the arguments $\overline{u}$ are reduced in one or more steps, including at least one $\to_{\mathcal{R}}$ step, otherwise this would be a pure $\to_{\mathcal{B}}$ derivation instead of a $\to_{\mathcal{R}} / \to_{\mathcal{B}}$ derivation. Eventually a reduction step must be given at the root with some rule $\mathsf{f}(\overline{w}) \to r_1 \in \mathcal{B}$. Now the infinite derivation continues from $r_1\sigma_1$, so there must be a minimal subterm of $r_1\sigma_1$ which starts an infinite derivation. By assumption the subterms $\overline{w}\sigma_1$ are all $(\to_{\mathcal{R}} / \to_{\mathcal{B}})$-terminating. Hence r is of the form $C[t_1]$ such that $t_1\sigma_1$ starts an infinite derivation. Because of the RHC condition we have that 1) $\mathsf{root}(t_1) \in \mathcal{F}_{\mathcal{B}}$, and 2) Every $\mathcal{R}$ redex in $r_1\sigma_1$ is rooted at some $p \in \mathcal{P}os_{\mathcal{V}}(r)$. That is, r_1 contributes no $\mathcal{R}$ redexes. The infinite derivation starting from $t_1\sigma_1$ contains strictly less $\mathcal{R}$ reduction steps than the infinite derivation starting from t_0.

This process can be repeated infinitely, obtaining always a new minimal non-terminating term rooted by a $\mathcal{B}$ symbol. As the number of $\mathcal{R}$ reduction steps in t_0 is finite, and every t_i contains strictly less $\mathcal{R}$ steps, eventually this becomes a pure $\mathcal{B}$ derivation, which contradicts the assumption. $\square$

Now we state the main result of this section, which is the generalization of Theorem 5.26 to relaxed hierarchical combinations.

Theorem 5.33 (Relative Termination Criterion(ii))**.** *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs such that $\mathcal{R} \cup \mathcal{B}$ is the RHC of the base $\mathcal{B}$ and the extension $\mathcal{R}$. Then, $\mathcal{R}$ terminates w.r.t. $\mathcal{B}$ if and only if there are no infinite $(DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$-chains.*

Proof. The proof scheme used in Theorem 5.26 suffices, replacing the uses of Lemma 5.25 with Lemma 5.32. $\square$

Using Theorem 5.33, we can produce a counterpart to Corollary 5.29 for goal-directed relative termination in relaxed hierarchical combinations.

Corollary 5.34 (Goal-Directed Relative Termination Criterion)**.** *Let $\mathcal{R}$ and $\mathcal{B}$ be two TRSs such that $\mathcal{R} \cup \mathcal{B}$ is the RHC of the base $\mathcal{B}$ and the extension $\mathcal{R}$, and let t_0 be an initial goal. Then, for all $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathcal{B}}$ derivation $\mathcal{D}$ starting from a term in $\lceil t_0 \rceil$, $\mathcal{D}$ is terminating if and only if there are no infinite $(t_0, DP(\mathcal{R}), \mathcal{R} \cup \mathcal{B})$-chains.*

5.3 Termination of Narrowing via Relative Termination

In this section, we consider the termination of *narrowing* and show how this problem can be reduced to proving the *relative* termination (see below) of a GTRS from an initial set of terms, putting to work the GDP framework introduced at the beginning of the chapter and the technique for proving relative termination introduced in the previous section.

Recently, [*Nishida and Vidal*, 2010; *Vidal*, 2008] introduced a termination analysis for narrowing which is roughly based on the following process[2]. First, following [*Antoy and Hanus*, 2006; *de Dios-Castro and López-Fraguas*, 2007], logic variables are replaced with a fresh function, called gen, which can be seen as a *data generator* that can be non-deterministically reduced to any ground (constructor) term. A first result relates the termination of narrowing in the original GTRS and the *relative* termination of rewriting using occurrences of gen to replace logic variables. However, in order to avoid dealing with relative termination, [*Vidal*, 2008] considers the use of an argument filtering to filter away occurrences of gen in the considered computations so that relative termination and termination coincide. Finally, termination is analyzed

[2]The termination analysis of logic programs of [*Schneider-Kamp et al.*, 2009b] follows a similar pattern but logic variables are replaced with *infinite* terms (the net effect, though, is similar).

using the DP framework [*Giesl et al.*, 2005c] for proving the termination of rewriting over the filtered terms.

This approach has several problems all related to the use of an argument filtering to filter the occurrences of gen. The most important one is that the search space for an argument filtering that allows to prove termination is exponential in the sum of the arities of the symbols in the signature, and moreover, this search cannot be casted as an optimization problem. This leads to the application of complex heuristics (as in [*Schneider-Kamp et al.*, 2009b]) which complicate the approach and diminish the effectiveness of the automation. Another issue is related to collapsing rules. Consider, for instance, a *collapsing* rule, i.e., a rule of the form $\mathsf{f}(x, y) \to y$, together with the argument filtering $\pi(\mathsf{f}) = \{1\}$. The filtered rule $\mathsf{f}(x) \to y$ contains an extra variable, y, and no refinement[3] of π will be able to eliminate it.[4]

Here, we argue that there is a better way to approach this problem. Instead of using a global argument filtering to filter away occurrences of gen, we propose to not filter them at all, and instead use the GDP framework developed in Section 5.1. As we show next this effectively solves the mentioned issues.

In order to formalize our approach, we first need to recall some existing notation and terminology from the literature.

Definition 5.35 ($\textsc{gen}_{\mathcal{C}}(\mathcal{R})$)**.** *Given a left-linear constructor GTRS $\mathcal{R}$ over the signature $\mathcal{F} = \mathcal{D} \uplus \mathcal{C}$, we define the generator of $\mathcal{R}$, $\textsc{gen}_{\mathcal{C}}(\mathcal{R})$, as the following set of rules:*

$$\textsc{gen}_{\mathcal{C}}(\mathcal{R}) \;=\; \{\;\; \mathsf{gen} \to \mathsf{c}(\overbrace{\mathsf{gen}, \ldots, \mathsf{gen}}^{n \text{ times}}) \;\mid\; \mathsf{c}/n \in \mathcal{C},\;\; n \geqslant 0 \;\;\}$$

Following [*Vidal*, 2008], $\widehat{t}$ denotes the result of replacing the variables of a term t by generators.

Definition 5.36 (gen-abstraction $\widehat{t}$)**.** *Given a term $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$, we let $\widehat{t} = t\sigma$, with $\sigma = \{x \mapsto \mathsf{gen} \mid x \in \mathcal{V}ar(t)\}$. Given a GTRS $\mathcal{R}$ we denote by $\widehat{\mathcal{R}}$ the result of replacing every extra variable in $\mathcal{R}$ (if any) with gen.*

Note that $\widehat{t}$ is ground for any term t since all variables occurring in t are replaced by the function gen. As for $\widehat{\mathcal{R}}$, we note that it contains no extra variables by definition.

We have that $\to_{\textsc{gen}_{\mathcal{C}}(\mathcal{R})}$ derivations are able to synthetize constructor terms.

Lemma 5.37. *Let $\mathcal{R}$ be a TRS defined over a signature $\Sigma = \mathcal{C} \uplus \mathcal{D}$, $t \in \mathcal{T}(\Sigma, \mathcal{V})$ be a term, and σ be a constructor substitution over Σ. Then $\widehat{t} \to^*_{\textsc{gen}_{\mathcal{C}}(\mathcal{R})} \widehat{t\sigma}$.*

Proof. By the definition of gen we know that $\mathsf{gen} \to^*_{\textsc{gen}_{\mathcal{C}}(\mathcal{R})} \widehat{c}$ for any $c \in \mathcal{T}(\mathcal{C})$, or equivalently $\widehat{x} \to^*_{\textsc{gen}_{\mathcal{C}}(\mathcal{R})} \widehat{c}$ for any $x \in \mathcal{V}$ and $c \in \mathcal{T}(\mathcal{C})$. Since the rewriting relation is closed under contexts, and σ is a constructor substitution, it follows that $\widehat{t} \to^*_{\textsc{gen}_{\mathcal{C}}(\mathcal{R})} \widehat{t\sigma}$. □

[3]An argument filtering π' is a refinement of another argument filtering π if it filters the same or more arguments, i.e., either $\pi'(\mathsf{f}) = \pi(\mathsf{f})$ or $\pi'(\mathsf{f}) \subseteq \pi(\mathsf{f})$ for every f.

[4]This is not a limitation of [*Schneider-Kamp et al.*, 2009b] since the considered rewrite systems that are produced from the translation of logic programs never have collapsing rules.

The completeness of replacing logic variables by generators is stated in the following result. This is a slight generalization of Lemma 1 in [Nishida and Vidal, 2010] to the case of TRSs with extra variables.

Theorem 5.38 (Completeness)**.** *Let $\mathcal{R}$ be a left-linear constructor GTRS over a signature $\Sigma = \mathcal{D} \uplus \mathcal{C}$ and $s \in \mathcal{T}(\Sigma, \mathcal{V})$ be a term. If $s \leadsto_{\sigma,\mathcal{R}} t$, then $\widehat{s} \overset{*}{\rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})}} \widehat{t}$.*

The statement is a slight variation of the completeness lemmas in [Antoy and Hanus, 2006]. Before giving a proof, we need to recall an auxiliary result from the literature.

Lemma 5.39 ([Nishida and Vidal, 2010])**.** *Let $\Sigma = \mathcal{C} \uplus \mathcal{D}$ be a signature and $t = \mathtt{f}(t_1, \ldots, t_n)$ be a linear term with $\mathtt{f} \in \mathcal{D}$ and $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{C}, \mathcal{V})$. For any term $s \in \mathcal{T}(\Sigma, \mathcal{V}) \setminus \mathcal{V}$ with $\mathcal{V}ar(t) \cap \mathcal{V}ar(s) = \emptyset$, it always holds that $\mathtt{mgu}(t, s)\!\upharpoonright_s$[5] is a constructor substitution.*

With the above lemma it is trivial to show that the substitution computed by a narrowing step in a left-linear constructor system is always a constructor substitution, allowing us to proceed with the proof of Theorem 5.38.

Proof of Theorem 5.38. We first prove a one-step version of the claim:

$$\text{If } s \overset{p}{\leadsto}_{\sigma,\mathcal{R}} t, \text{then } \widehat{s} \rightarrow^*_{\mathrm{GEN}_C(\mathcal{R})} \widehat{s\sigma} \rightarrow_{\widehat{\mathcal{R}}} \widehat{t} \tag{5.9}$$

The first segment, $\widehat{s} \rightarrow^*_{\mathrm{GEN}_C(\mathcal{R})} \widehat{s\sigma}$, follows from Lemma 5.37 and the auxiliar Lemma 5.39 introduced above.

Let $l \rightarrow r \in \mathcal{R}$ be the rule used in the narrowing step. Then $s\sigma = l\sigma$ with $\sigma = mgu(s, l)$, and $t = s\sigma[r\sigma]_p$. By assumption there must be a rule $l \rightarrow r\gamma \in \widehat{\mathcal{R}}$ with $\gamma = \{x \mapsto \mathsf{gen} \mid x \in \mathcal{EV}ar(l \rightarrow r)\}$. Therefore $s\sigma \rightarrow_{l \rightarrow r\gamma} s\sigma[r\gamma\sigma]_p$. By definition, $\widehat{s\sigma} = s\sigma\gamma'$, where γ' is the substitution mapping every variable in $\mathcal{V}ar(s\sigma)$ to gen. Closedness under instantiation of the rewriting relation yields $\widehat{s\sigma} \rightarrow_{l \rightarrow r\gamma} s\sigma\gamma'[r\gamma\sigma\gamma']_p$.

It remains to be shown that the term obtained, $s\sigma\gamma'[r\gamma\sigma\gamma']_p$, is equal to $\widehat{t}$. By Definition 5.36, $\widehat{t} = \widehat{s\sigma}[\widehat{r\sigma}]_p = s\sigma\gamma''[r\sigma\gamma'']_p$, where γ'' maps all the variables in $s\sigma$ and all the extra variables in r to gen. Hence, $\gamma'' = \gamma\gamma'$. Because σ is a most general unifier, for all $x \in \mathcal{EV}ar(l \rightarrow r)\}$, $x \notin D(\sigma)$. Therefore $D(\gamma) \cap D(\sigma) = \emptyset$ which yields $\gamma\sigma = \sigma\gamma$, which in turn implies $r\gamma\sigma\gamma' = r\sigma\gamma\gamma'$. This concludes the proof for the one-step version (5.9).

The original claim follows now easily by induction on the length of the derivation. The base case, a derivation with zero steps, holds trivially. For the inductive case we consider a derivation of $1 + n$ steps of the form $s \overset{p}{\leadsto}_{\sigma_1,\mathcal{R}} s_1 \leadsto^*_{\mathcal{R},\sigma_2} t$. Claim (5.9) yields $\widehat{s} \rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})} \widehat{s_1}$ for the first step, and the induction hypothesis provides $\widehat{s_1} \overset{*}{\rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})}} \widehat{t}$, completing the proof. $\square$

In [Vidal, 2008], the (possibly infinite) set of initial terms T was described by means of an *abstract goal*.

[5]We assume that the unification algorithm produces idempotent substitutions.

Definition 5.40 (Abstract Goal [Vidal, 2008]). *Let $\Sigma = \mathcal{D} \uplus \mathcal{C}$ be a signature. An abstract goal over Σ has the form $\mathsf{f}(m_1, \dots, m_n)$, where $\mathsf{f} \in \mathcal{D}$ and m_i is either $\mathbf{g}$ (ground constructor term) or $\mathbf{v}$ (constructor term possibly with variables), for all $i = 1, \dots, n$.*

Any abstract goal implicitly induces a (possibly infinite) set of terms, namely the concretization of t^α, in symbols $\gamma(t^\alpha)$, obtained as follows:

$$\gamma(\mathsf{f}(m_1 \dots m_n)) = \{\mathsf{f}(t_1 \dots t_n) \mid \begin{array}{ll} t_i \in \mathcal{T}(\mathcal{C}) & \text{if } m_i = \mathbf{g}, \\ t_i \in \mathcal{T}(\mathcal{C} \cup \mathcal{V}) & \text{if } m_i = \mathbf{v}\} \end{array}$$

Then, [Vidal, 2008; Nishida and Vidal, 2010] show that the termination of narrowing can be recast in terms of the relative termination of rewriting. The theorem below is a slight generalization to TRSs with extra variables. In the following, we say that a set of terms T is $\rightsquigarrow_{\mathcal{R}}$-terminating if there is no term $t_1 \in T$ such that an infinite sequence of the form $t_1 \rightsquigarrow_{\mathcal{R}} t_2 \rightsquigarrow_{\mathcal{R}} \dots$ exists.

Theorem 5.41 (Termination of Narrowing). *Let $\mathcal{R}$ be a left-linear constructor GTRS and t^α be an abstract goal. Then, $\gamma(t^\alpha)$ is $\rightsquigarrow_{\mathcal{R}}$-terminating if $\widehat{\gamma(t^\alpha)}$ is $\rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})}$ terminating.*

Proof. Assume that there is an infinite $\rightsquigarrow_{\mathcal{R}}$ derivation starting from a term $t_1 \in \widehat{\gamma(t^\alpha)}$, of the form $t_1 \rightsquigarrow_{\mathcal{R}} t_2 \rightsquigarrow_{\mathcal{R}} \dots$. Then by Theorem 5.38 there is a derivation $t_1 \rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})} t_2 \rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})} \dots$, which is also infinite, proving the claim. □

5.3.1 Beyond left-linear systems

A novel contribution of this thesis is to show that the left-linearity restriction can actually be lifted. For this some changes are required to the construction of the generator function. Concretely, the generator must be extended to generate not only every possible constructor term, but also every possible normal form. The resulting rewriting with generators relation does not simulate $\rightsquigarrow_{\mathcal{R}}$ derivations faithfully[6]. This is not a problem, since we are interested in termination, and fortunately termination of the resulting relation still implies termination of $\rightsquigarrow_{\mathcal{R}}$ as in the case of left–linear systems. The following example motivates the need to introduce of new generator rules when the system is not left-linear.

Example 5.42. *Let $\mathcal{R}$ be the TRS defined by the rules:*

$$\begin{aligned} \mathtt{f}(x, x) &\rightarrow \mathtt{f}(\mathtt{g}(\mathtt{0}), x) \\ \mathtt{g}(\mathtt{1}) &\rightarrow \mathtt{1} \end{aligned}$$

Given the initial goal $\mathtt{f}(x, y)$, we have an infinite ($\rightsquigarrow_{\mathcal{R}}$) derivation:

$$\underline{\mathtt{f}(x, y)} \rightsquigarrow_{\mathcal{R}} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}), x)} \rightsquigarrow_{\mathcal{R}} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}), \mathtt{g}(\mathtt{0}))} \rightsquigarrow_{\mathcal{R}} \dots$$

[6]Although it does faithfully simulate derivations where the substitutions computed by $\rightsquigarrow_{\mathcal{R}}$ are normalized, but this is not relevant for our purposes.

However there is no infinite $\rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\mathrm{GEN}_C(\mathcal{R})}$ *derivation starting from* $\widehat{\mathtt{f}(x, y)}$*:*

$$\mathtt{f}(\underline{\mathsf{gen}}, \mathsf{gen}) \rightarrow_{\mathcal{R} \cup \mathrm{GEN}_C(\mathcal{R})} \mathtt{f}(\mathsf{g}(\mathsf{0}), \mathsf{gen}) \rightarrow^*_{\mathcal{R} \cup \mathrm{GEN}_C(\mathcal{R})} \circ \not\rightarrow_{\mathcal{R}}$$

since there is no reduction from gen *to* g(0) *for the corresponding definition of* gen *to this TRS:*

$$\mathsf{gen} \rightarrow \mathsf{0}$$
$$\mathsf{gen} \rightarrow \mathsf{1}$$

In the example, gen is not able to synthetize the normal form g(0) because it is not a constructor term. Loosely speaking, in a constructor TRS we have a non constructor normal form $\mathtt{f}(t_1, \ldots, t_n)$ when $t_1 \ldots t_n$ do not belong to the domain of f, when seeing rules as a means of defining relations between sets of terms. That is, if the relation induced by the rules of f is not completely defined, then there are non constructor normal forms rooted by f.

In the case of left-linear constructor TRSs, non constructor normal forms are not observable since they cannot be pattern matched by the left-hand side of a left-linear rule. However as Example 5.42 shows, this is not true for non-left-linear rules. As completely defined rules are not a requirement for termination, in absence of left-linearity we must extend the definition of gen so that it covers these normal forms too. We introduce an additional set of rules $\mathrm{GEN}_{\mathrm{NF}}(\mathcal{R})$:

Definition 5.43 ($\mathrm{GEN}_{\mathrm{NF}}(\mathcal{R})$)**.** *Let* $\mathcal{R}$ *be a GTRS over a signature* $\Sigma = \mathcal{C} \uplus \mathcal{D}$ *such that* gen $\notin \Sigma$*. We define the (usually infinite) set of root-defined, ground normal form generators:*

$$\mathrm{GEN}_{\mathrm{NF}}(\mathcal{R}) = \{\mathsf{gen} \rightarrow t \mid \mathtt{f}/n \in \mathcal{D}, t = \mathtt{f}(t_0, \ldots, t_n) \in \mathcal{T}(\Sigma) \cap \mathrm{NF}(\mathcal{R})\}$$

The union of the constructor generators plus the normal form generators is denoted as $\mathrm{GEN}(\mathcal{R})$*:*

$$\mathrm{GEN}(\mathcal{R}) = \mathrm{GEN}_C(\mathcal{R}) \cup \mathrm{GEN}_{\mathrm{NF}}(\mathcal{R})$$

Although the definition of $\mathrm{GEN}_{\mathrm{NF}}(\mathcal{R})$ is not finitely computable –as it contains an infinite number of rules as well as possibly infinite terms in the right-hand sides– we will show later that this does not constitute a problem when proving $\rightarrow_{\mathcal{R}} / \rightarrow_{\mathrm{GEN}(\mathcal{R})}$ termination.

For the TRS of Example 5.42, the infinite set $\mathrm{GEN}_{\mathrm{NF}}(\mathcal{R})$ includes the rules:

$$\begin{aligned}
&\mathsf{gen} \rightarrow \mathsf{g}(\mathsf{0})\\
&\mathsf{gen} \rightarrow \mathsf{g}(\mathsf{g}(\mathsf{0}))\\
&\mathsf{gen} \rightarrow \mathsf{g}(\mathsf{g}(\mathsf{g}(\ldots)))\\
&\mathsf{gen} \rightarrow \mathtt{f}(\mathsf{0}, \mathsf{1})\\
&\mathsf{gen} \rightarrow \mathtt{f}(\mathsf{1}, \mathsf{0})\\
&\mathsf{gen} \rightarrow \mathtt{f}(\mathsf{g}(\mathsf{0}), \mathsf{g}(\mathsf{1}))
\end{aligned}$$

$$\begin{aligned}
&\mathtt{gen} \to \mathtt{f}(\mathtt{g}(\mathtt{1}), \mathtt{g}(\mathtt{0}))\\
&\mathtt{gen} \to \mathtt{f}(\mathtt{0}, \mathtt{g}(\dots))\\
&\mathtt{gen} \to \mathtt{f}(\mathtt{g}(\dots), \dots)\\
&\mathtt{gen} \to \mathtt{f}(\mathtt{0}, \mathtt{f}(\dots, \dots))\\
&\mathtt{gen} \to \mathtt{f}(\mathtt{g}(\dots), \mathtt{f}(\dots, \dots))\\
&\quad .\\
&\quad .\\
&\quad .
\end{aligned}$$

With the new definition of $\textsc{gen}(\mathcal{R})$, we generalize Lemma 5.37 to non left–linear systems.

Lemma 5.44. *Let $\mathcal{R}$ be a TRS over a signature $\Sigma = \mathcal{C} \uplus \mathcal{D}$, $t \in \mathcal{T}(\Sigma, \mathcal{V})$ be a term and σ be a normalized substitution. Then $\widehat{t} \to^*_{\textsc{gen}(\mathcal{R})} \widehat{t\sigma}$.*

Proof. By the definition of $\mathtt{gen}$, it is clear that $\mathtt{gen} \to^*_{\textsc{gen}(\mathcal{R})} \widehat{t}$ for any $t \in \textsc{nf}(\mathcal{R})$, or equivalently $\widehat{x} \to^*_{\textsc{gen}(\mathcal{R})} \widehat{t}$ for any $x \in \mathcal{V}$ and $t \in \textsc{nf}(\mathcal{R})$. Since the rewriting relation is closed under contexts and σ is a normalized substitution, it follows that $\widehat{t} \to^*_{\textsc{gen}(\mathcal{R})} \widehat{t\sigma}$. □

We take a small detour now to state a result about the commutation of a special rewriting relation, the parallel reductions relation. We prove a version of the well-known parallel moves lemma of [*Huet*, 1980], which then enables us to prove a result on the completeness of simulating $\rightsquigarrow_{\mathcal{R}}$ with $\to_{\mathcal{R} \uplus \textsc{gen}(\mathcal{R})}$ in non left–linear systems.

The original formulation of the parallel moves lemma assumes left-linearity; we prove it here for non left-linear systems by restricting the available parallel positions used in every step. First recall the parallel reduction relation.

Definition 5.45 (Parallel Reduction Relation [*Baader and Nipkow*, 1998, Definition 6.3.7]). *Let $\mathcal{R}$ be a TRS and $P = p_1 \dots p_n$ be a possibly empty set of pairwise disjoint positions, each one associated to a rule $l_{p_i} \to r_{p_i} \in \mathcal{R}$. We write $s \overset{P}{\rightrightarrows}_{\mathcal{R}} t$ if $s \xrightarrow{p_1}_{l_{p_1} \to r_{p_1}} \circ \xrightarrow{p_2}_{l_{p_2} \to r_{p_2}} \circ \dots \xrightarrow{p_n}_{l_{p_n} \to r_{p_n}} t$. The decorations P and $\mathcal{R}$ can be dropped if they are obvious from the context or irrelevant.*

In order to extend the parallel moves lemma of [*Huet*, 1980] to non-left-linear systems, we demand that all reducts corresponding to a repeated variable are reduced in parallel, and using the same rule. This ensures that the reduction never gets "stuck", and therefore both reduction paths are able to reach the same normal form.

Lemma 5.46 (Parallel Moves for non-left-linear systems)**.** *Let $\mathcal{R}$ be a TRS over a signature $\Sigma = \mathcal{C} \uplus \mathcal{D}$, $l \to r \in \mathcal{R}$ be a rule, σ be a substitution and $P \subseteq \mathcal{P}os_{\mathcal{D}}(l\sigma)$ be a pairwise disjoint set of positions such that for all $p \in P$, there is some $p' \in \mathcal{P}os_{\mathcal{V}}(l)$ with $p' \leq p$. That is, every position in P is below some variable position of l.*

Then, if $l\sigma \overset{P}{\rightrightarrows}_{\mathcal{R}} l\sigma'$, there exists a substitution σ' such that $r\sigma \to_{\mathcal{R}} r\sigma' \leftleftarrows_{\mathcal{R}} l\sigma'$, as summarized in Figure 5.1.

$$\begin{tikzcd}
l\sigma \arrow[r] \arrow[d, shift left=1] \arrow[d, shift right=1] & r\sigma \arrow[d, dotted, shift left=1] \arrow[d, dotted, shift right=1] \\
l\sigma' \arrow[r, dotted] & r\sigma'
\end{tikzcd}$$

Figure 5.1: The parallel moves lemma for non-left-linear systems

Remark 5.47. *There is an implicit condition in the above formulation which replaces the requirement of left-linearity. Namely, we require that $l\sigma$ is reduced in parallel to a term $l\sigma'$. Since l is not linear, this imposes some conditions on the construction of P. Concretely, all the copies of a redex below an occurrence of a repeated variable are reduced in parallel using the same rule. More formally:*

- *For every variable occurring more than once in l at positions $p_1 \dots p_n$, then for any position q, either $p_i.q \notin P$ for all i, or $p_i.q \in P$ for all i and $l_{p_1.q} \to r_{p_1.q} = l_{p_2.q} \to r_{p_2.q} = \dots$*

Before giving the proof of Lemma 5.46, we recall the following auxiliary lemma from [Baader and Nipkow, 1998]. The proof is omitted since it can be found in the source.

Lemma 5.48 ([Baader and Nipkow, 1998, Lemma 6.4.2])**.** *If $x\sigma \rightrightarrows x\sigma'$ for all $x \in Var(s)$, then $s\sigma \rightrightarrows s\sigma'$*

We are now ready to proceed with the proof of the parallel moves lemma for non-left-linear systems.

Proof of Lemma 5.46. This generalization of the parallel moves lemma to non-left linear TRSs should follow trivially from the original formulation. Our proof is based on the proof found in [Baader and Nipkow, 1998].

Define the set P_x of the redex positions below an occurrence of x:

$$P_x = \{q \in \mathcal{P}os(x\sigma) \mid q_x \in \mathcal{P}os_x(l), q_x.q \in P\}$$

Thus, for all $p \in P_x$, there is a rule $l_p \to r_p \in \mathcal{R}$ and a substitution σ_p such that $(x\sigma)|_p = l_p\sigma_p$.

Now define the substitution $\sigma'(x) = (x\sigma)[r_p\sigma_p]_{p \in P_x}$ for $x \in Var(l)$. By definition we have that $x\sigma \rightrightarrows x\sigma'$. By Lemma 5.48 above, this implies $r\sigma \rightrightarrows r\sigma'$.

On the other hand, clearly $l\sigma' \to_{l \to r} r\sigma'$, and we are done. □

Thanks to the parallel moves lemma, we are able to generalize the completeness result of Theorem 5.38. As rewriting with generators does not simulate narrowing anymore, the generalized result simply states that every narrowing derivation has an associated rewriting with generators derivation.

$$\begin{tikzcd}
 & s\sigma \arrow[r, "p"] \arrow[d, two heads, shift left=0.5ex] \arrow[d, two heads, shift right=0.5ex, "{>p}"'] & t\sigma \arrow[d, dotted, two heads, shift left=0.5ex] \arrow[d, dotted, two heads, shift right=0.5ex] \\
 & {} \arrow[r, dotted] \arrow[d, two heads, shift left=0.5ex] \arrow[d, two heads, shift right=0.5ex, "{>p}"'] & {} \arrow[d, dotted, two heads, shift left=0.5ex] \arrow[d, dotted, two heads, shift right=0.5ex] \\
 & s\sigma[s'_p\sigma']_p \arrow[r, dotted] \arrow[d, dotted] & s\sigma[r\sigma']_p \arrow[d, dotted] \\
\widehat{s\sigma'} \arrow[r, "{>p}", "*"'] & \widehat{s\sigma'}[\widehat{s'_p\sigma'}]_p \arrow[r, dotted, "p"] & \widehat{s\sigma'}[\widehat{r\sigma'}]_p = \widehat{t\sigma'}
\end{tikzcd}$$

Figure 5.2: Proof of Theorem 5.49

Theorem 5.49. *Let $\mathcal{R}$ be a constructor TRS over a signature $\Sigma = \mathcal{D} \uplus \mathcal{C}$ and $s \in \mathcal{T}(\Sigma, \mathcal{V})$ be a term. Then, if $s \rightsquigarrow^*_{\sigma,\mathcal{R}} t\sigma$ and every term in the range of σ has a normal form, there is a $\sigma' = \sigma\!\downarrow$ such that $\widehat{s} \overset{*}{\to_{\widehat{\mathcal{R}}}/\to_{\text{GEN}(\mathcal{R})}} \widehat{t\sigma'}$.*

Proof. We prove the claim by performing induction on the length of the narrowing derivation. The base case, for a derivation of length zero, is trivial. For the induction case, assume a derivation of $n+1$ steps, of the form $s \rightsquigarrow^*_{\sigma,\mathcal{R}} u\sigma \overset{p}{\rightsquigarrow}_{\theta,l\to r} t\sigma\theta$. By the induction hypothesis, the n steps derivation $s \rightsquigarrow^*_{\sigma,\mathcal{R}} u\sigma$ yields

$$\widehat{s} \overset{*}{\to_{\widehat{\mathcal{R}}}/\to_{\text{GEN}(\mathcal{R})}} \widehat{u\sigma'}$$

with $\sigma' = (\sigma\!\restriction_u)\!\downarrow$ a normal form of σ. We prove the claim by showing that the rest of the derivation can be constructed, i.e. the segment

$$\widehat{u\sigma'} \overset{*}{\to_{\widehat{\mathcal{R}}}/\to_{\text{GEN}(\mathcal{R})}} \widehat{t\sigma'\theta'}$$

Lemma 5.39 yields $\widehat{u\sigma'} \to^*_{\text{GEN}(\mathcal{R})} \widehat{u\sigma'\theta'}$. In the following, let $l \to r\gamma' \in \widehat{\mathcal{R}}$ be the rule corresponding to $l \to r \in \mathcal{R}$, where γ' is a concretization substitution replacing variables by calls to gen. Also, let us write $\widehat{u\sigma'\theta'}$ as $u\sigma'\theta'\gamma[u_p\sigma'\theta'\gamma]_p$, and $\widehat{t\sigma'\theta'}$ as $u\sigma'\theta'\gamma[r\gamma'\sigma'\theta'\gamma]_p$. We can ignore the context $u\sigma'\theta'$, as well as the concretization substitutions γ, γ' and the concretization of $\mathcal{R}$, $\widehat{\mathcal{R}}$. It suffices to prove $u_p\sigma'\theta' \to^*_{\mathcal{R}} r\sigma'\theta'$, and from that, the claim is derived by monotonicity and stability of the rewriting relation.

By the narrowing step, θ is a mgu of $u_p\sigma$ and l, so $u_p\sigma\theta = l\theta$. Hence $\theta\!\restriction_l$ is a matcher of $u_p(\sigma\theta)\!\restriction_{u_p}$ and l. Let $l\theta\!\restriction_l = u_p(\sigma\theta)\!\restriction_{u_p} \rightrightarrows^*_{\widehat{\mathcal{R}}} u'_p(\sigma'\theta')\!\restriction_{u'_p} = l\theta'\!\restriction_l$ be the derivation which normalizes $(\sigma\theta)\!\restriction_{u_p}$ while at the same time satisfying the parallel reductions condition for non-left-linear systems (cf. Remark 5.47), i.e. it reduces every occurrence of a duplicated variable of l in parallel. Because of the parallel restriction, some steps in the derivation may occur inside u_p, and hence we speak of u'_p. And thanks to the parallel restriction, and the fact that all the subterms of l are constructor terms, it is clear that we can write $u'_p(\sigma'\theta')\restriction_{u'_p}$ as $l\theta'\restriction_l$. The parallel moves lemma yields $l\theta'\restriction_l = u'_p(\sigma'\theta')\restriction_{u'_p} \overset{\epsilon}{\rightarrow}_{l\to r} r\sigma'\theta'$.

Recall that we start from $u_p\sigma'\theta'$. From the parallel derivation above, and monotonicity, it follows that there is a derivation $u_p\sigma'\theta' \overset{>\epsilon *}{\rightarrow}_{\widehat{\mathcal{R}}} u'_p\sigma'\theta' \overset{\epsilon}{\rightarrow}_{\mathcal{R}} r\theta'$, By monotonicity and stability, this can be lifted to $u\sigma'\theta'\gamma[u_p\sigma'\theta'\gamma]_p \overset{>p*}{\rightarrow}_{\widehat{\mathcal{R}}} u\sigma'\theta'\gamma[u'_p\sigma'\theta'\gamma]_p \overset{p}{\rightarrow}_{l\to r\gamma'} u\sigma'\theta'\gamma[r\gamma'\theta'\gamma]_p$, or simply $\widehat{u\sigma'\theta'} \rightarrow^*_{\widehat{\mathcal{R}}} \widehat{t\sigma'\theta'}$. This is summarized in Figure 5.2 $\square$

Example 5.50. *Consider again the TRS $\mathcal{R}$ of Example 5.42:*

$$\begin{aligned} \mathtt{f}(x,x) &\to \mathtt{f}(\mathtt{g}(\mathtt{0}),x) \\ \mathtt{g}(\mathtt{1}) &\to \mathtt{1} \end{aligned}$$

Let $t_0 = \mathtt{f}(x,y)$. Given the infinite ($\rightsquigarrow_{\mathcal{R}}$) derivation starting from t_0:

$$\underline{\mathtt{f}(x,y)} \rightsquigarrow_{\mathcal{R},\{y\mapsto x\}} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}),x)} \rightsquigarrow_{\mathcal{R},\{x\mapsto \mathtt{g}(\mathtt{0})\}} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}),\mathtt{g}(\mathtt{0}))} \rightsquigarrow_{\{\},\mathcal{R}} \cdots$$

the $\rightarrow_{\widehat{\mathcal{R}}=\mathcal{R}}/\overset{*}{\rightarrow}_{GEN(\mathcal{R})}$ derivation that mirrors it from $\widehat{t_0}$ is:

$$\underline{\mathtt{f}(\mathtt{gen},\mathtt{gen})} \rightarrow^*_{\mathrm{GEN}(\mathcal{R})} \underline{\mathtt{f}(\mathtt{gen},\mathtt{gen})} \rightarrow^*_{\widehat{\mathcal{R}}} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}),\mathtt{gen})} \rightarrow^*_{\mathrm{GEN}(\mathcal{R})} \underline{\mathtt{f}(\mathtt{g}(\mathtt{0}),\mathtt{g}(\mathtt{0}))} \rightarrow^*_{\widehat{\mathcal{R}}} \cdots$$

Example 5.51. *Consider the TRS $\mathcal{R}$ given by the rules:*

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

and the initial term $t_0 = \mathtt{c}(\mathtt{f}(\mathtt{a},x),\mathtt{f}(x,\mathtt{a}))$.

The following $\rightsquigarrow_{\mathcal{R}}$ derivation starting from t_0:

$$\mathtt{c}(\underline{\mathtt{f}(\mathtt{a},x)},\mathtt{f}(x,\mathtt{a})) \rightsquigarrow_{\mathcal{R},\{x\mapsto \mathtt{a}\}} \mathtt{c}(\mathtt{a},\underline{\mathtt{f}(\mathtt{a},\mathtt{a})}) \rightsquigarrow_{\mathcal{R},\{\}} \mathtt{c}(\mathtt{a},\mathtt{a})$$

is simulated by the following ($\widehat{\mathcal{R}} = \mathcal{R}$) $\rightarrow_{\mathcal{R}}/\overset{*}{\rightarrow}_{GEN(\mathcal{R})}$:

$$\begin{aligned} &\mathtt{c}(\mathtt{f}(\mathtt{a},\underline{\mathtt{gen}}),\mathtt{f}(\underline{\mathtt{gen}},\mathtt{a})) \rightarrow^*_{\mathrm{GEN}(\mathcal{R})} \mathtt{c}(\mathtt{f}(\underline{\mathtt{a}},\mathtt{b}),\mathtt{f}(\mathtt{b},\mathtt{a})) \rightarrow_{\mathcal{R}} \\ &\qquad \mathtt{c}(\underline{\mathtt{f}(\mathtt{b},\mathtt{b})},\mathtt{f}(\mathtt{b},\mathtt{a})) \rightarrow_{\mathcal{R}} \mathtt{c}(\mathtt{b},\mathtt{f}(\mathtt{b},\underline{\mathtt{a}})) \rightarrow_{\mathcal{R}} \mathtt{c}(\mathtt{b},\underline{\mathtt{f}(\mathtt{b},\mathtt{b})}) \rightarrow_{\mathcal{R}} \mathtt{c}(\mathtt{b},\mathtt{b}) \end{aligned}$$

Thanks to Theorem 5.49, termination of narrowing in non left–linear systems can also be shown via relative termination of rewriting. The key insight, which comes from the results in Chapter 4, is that minimal infinite narrowing derivations in constructor systems are always of the TRAT kind, as per the classification of Lemma 4.15, and hence there are no HYBRID steps which render Theorem 5.49 unapplicable.

Theorem 5.52 (Termination of Narrowing(ii)). *Let $\mathcal{R}$ be a constructor GTRS and t^α be an abstract goal. Then, $\gamma(t^\alpha)$ is $\rightsquigarrow_{\mathcal{R}}$-terminating if $\widehat{\gamma(t^\alpha)}$ is $(\rightarrow_{\widehat{\mathcal{R}}} / \rightarrow_{\text{GEN}(\mathcal{R})})$-terminating.*

Proof. Assume that there is an infinite $\rightsquigarrow_{\mathcal{R}}$ derivation starting from a term $t_1 \in \gamma(t^\alpha)$:

$$t_1 \rightsquigarrow_{\sigma_1,\mathcal{R}} t_2 \rightsquigarrow_{\sigma_2,\mathcal{R}} t_3 \rightsquigarrow_{\sigma_3,\mathcal{R}} \cdots$$

W.l.o.g. we assume that t_1 is a minimal non-terminating term, i.e. $t_1 \in \mathcal{T}^\infty_{\mathcal{R}}$. By Corollary 4.33, we know that $\rightsquigarrow_{\mathcal{R}}$ enjoys the TRAT property (cf. Definition 4.9). Proposition 4.31 ensures that this infinite derivation is composed solely of TRAT steps. By the classification of Lemma 4.15, we see that every σ_i computed in this narrowing derivation has the property that all its bindings have normal forms, because:

- Every binding coming from a proper subterm of the term being narrowed is $(\rightsquigarrow_{\mathcal{R}})$-terminating by assumption, and
- Every binding coming from a proper subterm of the left-hand side is a constructor term instantiated with terminating terms. This is a $(\rightsquigarrow_{\mathcal{R}})$-terminating term, as there are no ECHOING steps.

Therefore, because every computed binding is normalizable, Theorem 5.49 can be applied in every step.

$P : s_1 \rightsquigarrow_{\mathcal{R}} s_2 \rightsquigarrow_{\mathcal{R}} \ldots$ starting from a minimal term $s_1 \in \mathit{calls}_{\mathcal{R}}(t_1)$. By Corollary 4.33, $\rightsquigarrow_{\mathcal{R}}$ enjoys the TRAT property, so eventually there is a top step in the derivation P and we have $P : s_1 \overset{>\epsilon *}{\rightsquigarrow}_{\mathcal{R},\sigma_1} s_2\sigma_1 \overset{\epsilon}{\rightsquigarrow}_{l\to r\in\mathcal{R},\sigma_2} s_3 \ldots$. By minimality we can safely assume that all the bindings in the range of σ are terminating. Hence the next minimally non terminating term must be rooted in r_1, i.e. s_3 is of the form $(C'_1[s_4]_p)$ with $p \in \mathcal{P}os_{\mathcal{D}}(r_1)$. Since s_4 is a minimal term again, we can start over and apply the TRAT property to have that eventually there is a narrowing step at position p. This can be repeated an infinite number of times, and P is of the form:

$$P : s_1 \overset{>\epsilon *}{\rightsquigarrow}_{\mathcal{R},\sigma_1} s_2 \overset{\epsilon}{\rightsquigarrow}_{l_1\to r_1\in\mathcal{R},\sigma_2} C_1[s_3]_{p_1} \overset{>p_1 *}{\rightsquigarrow}_{\mathcal{R},\sigma_3} C'_1[s_4]_{p_1} \overset{p_1 *}{\rightsquigarrow}_{\mathcal{R},\sigma_4} \cdots$$

In every narrowing step in the derivation, the terms in the range of the computed substitution are terminating (and hence have normal forms). Hence Theorem 5.49 applies, and there is an infinite parallel rewriting derivation $s_1 \rightarrow_{\widehat{\mathcal{R}}}/\overset{*}{\rightarrow}_{\text{GEN}(\mathcal{R})}$. □

5.3.2 Checking the termination criterion

Theorems 5.41 and 5.52 provide a theoretical basis for proving the termination of narrowing from an initial goal in terms of the termination of relative rewriting with generators. From here, Corollary 5.34 provides a method for checking the termination of relative rewriting from an initial goal. There are mainly two practical obstacles in the way to obtain an effective method: 1) encoding abstract goals as initial goals, and 2) handling the infinite number of generator rules in the case of non-left-linear systems.

First, let us recall that our GDP problems consider an initial goal rather than an abstract goal. To overcome this minor difference, we embed the abstract goal into the TRS by means of an additional rule.

Definition 5.53 (Goal Rule)**.** *Let $\mathcal{R}$ be a TRS and $t^\alpha = \mathsf{f}(t_1,\dots,t_n)$ be an abstract goal with m occurrences of* $\mathbf{g}$. *We let $goal(t^\alpha)$ denote the rule*

$$\mathsf{goal}(x_{j_1},\dots,x_{j_m}) \to \mathsf{f}(x_1,\dots,x_n)$$

where $x_1,\dots,x_n$ are (fresh) distinct variables and $j_1,\dots,j_m$ are the positions of the $\mathbf{g}$ *arguments of t^α.*

Given a TRS $\mathcal{R}$ and an abstract goal t^α, we denote by $\mathcal{R}_{t^\alpha}$ the TRS that extends $\mathcal{R}$ with this rule; formally, $\mathcal{R}_{t^\alpha} = \mathcal{R} \cup \{goal(t^\alpha)\}$.

In other words, we replace the $\mathbf{v}$ arguments of the abstract goal t^α by extra variables. Extra variables occur very naturally in the context of narrowing since they behave as *free* variables which can only be instantiated to finite constructor terms. In our context they are simply replaced by occurrences of gen by the $\widehat{\ }$ operator.

The following result combining Corollaries 5.29 / 5.34 and Theorems 5.41 / 5.52, is the basis of our termination proving method.

Theorem 5.54. *Let $\mathcal{R}$ be a constructor GTRS over a signature $\Sigma = \mathcal{C} \uplus \mathcal{D}$, and t^α an abstract goal. Let $\mathsf{goal}(x_1,\dots,x_n)$ be the left-hand side of $goal(t^\alpha)$. $\gamma(t^\alpha)$ is $\leadsto_{\mathcal{R}}$-terminating when:*

- $(\mathsf{goal}(x_1,\dots,x_n), DP(\widehat{\mathcal{R}_{t^\alpha}}), \widehat{\mathcal{R}_{t^\alpha}} \cup \textsc{gen}_C(\mathcal{R}), \mathbf{a})$ *is finite, for $\mathcal{R}$ left-linear.*
- $(\mathsf{goal}(x_1,\dots,x_n), DP(\widehat{\mathcal{R}_{t^\alpha}}), \widehat{\mathcal{R}_{t^\alpha}} \cup \textsc{gen}(\mathcal{R}), \mathbf{a})$ *is finite, otherwise.*

Proof. Suppose that $\gamma(t^\alpha)$ is not $\leadsto_{\mathcal{R}}$-terminating. Then, there is a term t_0 in $\gamma(t^\alpha)$ starting an infinite narrowing derivation. Then by Theorem 5.41 (resp. Theorem 5.52 if $\mathcal{R}$ is not left-linear), t_0 is not $(\to_{\widehat{\mathcal{R}}} / \to_{\textsc{gen}_C(\mathcal{R})})$-terminating (resp. $(\to_{\widehat{\mathcal{R}}} / \to_{\textsc{gen}(\mathcal{R})})$-terminating).

Or equivalently, $\mathsf{goal}(t_1,\dots,t_n)$ is not $(\to_{\widehat{\mathcal{R}_{t^\alpha}}} / \to_{\textsc{gen}_C(\mathcal{R})})$-terminating (resp. $(\to_{\widehat{\mathcal{R}_{t^\alpha}}} / \to_{\textsc{gen}(\mathcal{R})})$-terminating) for some $t_1,\dots,t_n$ in $\mathcal{T}(\mathcal{C})$ such that $\mathsf{goal}(t_1,\dots, t_n) \to_{goal(t^\alpha)} t_0$.

Hence there is an infinite $\to_{\widehat{\mathcal{R}_{t^\alpha}}} / \to_{\textsc{gen}_C(\mathcal{R})}$ derivation (resp. an infinite $\to_{\widehat{\mathcal{R}_{t^\alpha}}} / \to_{\textsc{gen}(\mathcal{R})}$ derivation) starting from $\mathsf{goal}(t_1,\dots,t_n)$. Corollary 5.29 (resp. 5.34) yields an infinite $(t_0, DP(\mathcal{R}), \widehat{\mathcal{R}_{t^\alpha}} \cup \textsc{gen}_C(\mathcal{R})$-chain (resp. an infinite $(t_0, DP(\mathcal{R}), \widehat{\mathcal{R}_{t^\alpha}} \cup \textsc{gen}(\mathcal{R})$-chain), which means the associated GDP problem is not finite, concluding the proof. □

Example 5.55. *Consider the following TRS that is part of Example 5.22:*

$$\mathcal{R} = \left\{ \begin{array}{lcl} \mathsf{add}(\mathsf{s}(x), y) & \to & \mathsf{s}(\mathsf{add}(x,y)) \\ \mathsf{add}(\mathsf{zero}, y) & \to & y \end{array} \right\}$$

For the abstract goal $t^\alpha = \mathsf{add}(\mathbf{g},\mathbf{v})$, the initial GDP problem $(\mathsf{goal}(x), DP(\widehat{\mathcal{R}_{t_\alpha}}), \widehat{\mathcal{R}_{t_\alpha}} \cup \textsc{gen}(\mathcal{R}), \mathbf{a})$ is reduced to the finite GDP problem in Example 5.22. Therefore, $\gamma(\mathsf{add}(\mathbf{g},\mathbf{v}))$ is $\leadsto_{\mathcal{R}}$-terminating.

When $\mathcal{R}$ is not left–linear, we obtain a GDP problem with an infinite number of rules, but with still a finite number of pairs. Therefore the dependency graph processor of Theorem 5.15 is still applicable. Even more, the estimation of Definition 4.28 can be paired with Theorem 5.12 to compute an estimated dependency graph. However, due to the infinite number of rules, this may not be possible for other more precise estimations, as the following one.

Theorem 5.56 (Improved Estimated Dependency Graph[*Giesl et al.*, 2006b])**.** *Let $\mathcal{R}$ be a set of rules (possibly with extra variables), and $\mathcal{P}$ be a set of pairs (possibly with extra variables). The* $\textsc{icap}_\mathcal{R}$ *estimation function is defined inductively as follows:*

$$\begin{aligned}
\textsc{icap}_\mathcal{R}(x) &= x', \ \textit{for a fresh variable } x' \\
\textsc{icap}_\mathcal{R}(\mathtt{f}(t_1,\dots,t_n)) &= x', \ \textit{for a fresh variable } x' \\
&\quad \textit{if } l \textit{ and } \mathtt{f}(\textsc{icap}_\mathcal{R}(t_1),\dots,\textsc{icap}_\mathcal{R}(t_n)) \textit{ unify for some } l \to r \in \mathcal{R} \\
\textsc{icap}_\mathcal{R}(\mathtt{f}(t_1,\dots,t_n)) &= \mathtt{f}(\textsc{icap}_\mathcal{R}(t_1),\dots,\textsc{icap}_\mathcal{R}(t_n)), \ \textit{otherwise.}
\end{aligned}$$

The improved estimated dependency graph, where the nodes are the pairs of $\mathcal{P}$ and there is an edge from $s^\# \to t^\#$ to $u^\# \to v^\#$ iff $\textsc{icap}_\mathcal{R}(t)$ and u are unifiable, is an over-estimation of the exact dependency graph.

Because $\textsc{icap}_\mathcal{R}$ uses unification instead of the syntactic test done by the $\textsc{cap}$ of Definition 4.28, the resulting approximation of the dependency graph is more precise and allows to prove termination in more cases. Unfortunately the improved estimation is not computable if the number of rules is infinite, since there are an infinite number of unification tests to perform for every non trivial right-hand side.

However, it is easily noticed that the rules of $\textsc{gen}_{\textsc{nf}}(\mathcal{R})$ can be ignored when computing the simple estimated graph of Definition 4.28, since they add no new defined symbols to the signature. It turns out that they can be ignored when using the improved estimation too, since they add no new left-hand sides.

Theorem 5.57. *Let $\mathcal{R}$ be a constructor TRS, and let $(\mathcal{P}, \mathcal{R} \cup \textsc{gen}(\mathcal{R}), f)$ be a DP problem. Then the improved estimated dependency graph of $(\mathcal{P}, \mathcal{R} \uplus \textsc{gen}_C(\mathcal{R}) \uplus \textsc{gen}_{\textsc{nf}}(\mathcal{R}), f)$ coincides with the improved estimated dependency graph of $(\mathcal{P}, \mathcal{R} \uplus \textsc{gen}_C(\mathcal{R}), f)$.*

Proof. Let $\mathcal{G}_0$ be the graph estimated for the complete problem and $\mathcal{G}_1$ for the problem ignoring the rules of $\textsc{gen}_{\textsc{nf}}(\mathcal{R})$, and suppose that $\mathcal{G}_0$ containts an edge between two pairs $s \to t$, $u \to v$ which is not present in $\mathcal{G}_1$. Then $\textsc{icap}_{\mathcal{R}\cup\textsc{gen}_C(\mathcal{R})\cup\textsc{gen}_{\textsc{nf}}(\mathcal{R})}(t)$ and u unify, whereas $\textsc{icap}_{\mathcal{R}\cup\textsc{gen}_C(\mathcal{R})}(t)$ and u do not. From the definition of $\textsc{icap}$, it follows that this is only possible if there is some rule $l \to r \in \textsc{gen}_{\textsc{nf}}(\mathcal{R})$ such that for some subterm t' of t, $\textsc{icap}_\mathcal{R}(t')$ and l unify. But this is impossible because either $\mathcal{C} \neq \emptyset$ and then there is already a rule $\mathsf{gen} \to s \in \textsc{gen}_C(\mathcal{R})$, or $\mathcal{C} = \emptyset$ and then $\textsc{gen}_C(\mathcal{R}) = \textsc{gen}_{\textsc{nf}}(\mathcal{R}) = \emptyset$. Therefore $\textsc{icap}_{\mathcal{R}\cup\textsc{gen}_C(\mathcal{R})\cup\textsc{gen}_{\textsc{nf}}(\mathcal{R})}(t)$ and $\textsc{icap}_{\mathcal{R}\cup\textsc{gen}_C(\mathcal{R})}(t)$ coincide, and every edge in $\mathcal{G}_0$ is also an edge in $\mathcal{G}_1$. □

The infinite number of rules in $\textsc{gen}_{\textsc{nf}}(\mathcal{R})$ also poses a problem for the reduction pair processors of Theorems 5.16 and 5.21, since an infinite number of rules implies an infinite number of order constraints to be satisfied, which is obviously uncomputable. In order to overcome this issue, we overapproximate the infinite set of constraints induced by the rules of $\text{GEN}_{\text{NF}}(\mathcal{R})$ with a finite set of constraints $\mathcal{G}en_{\text{NF}}^{\succsim^\pi}$. Below we introduce a version of the reduction pair processor of Theorem 5.21 which handles the rules from $\text{GEN}_{\text{NF}}(\mathcal{R})$ in this special way.

Theorem 5.58. *Let $(\succsim, \succ)$ be a reduction pair and π be an argument filtering. Let $\mathcal{R} = \text{GEN}_{\mathcal{C}}(\mathcal{R}') \cup \mathcal{R}'$ be a non left-linear TRS. Given a GDP problem $(t_0, \mathcal{P}, \mathcal{R} \cup \text{GEN}_{\text{NF}}(\mathcal{R}')\}, f)$, Proc returns:*

- $(t_0, \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R} \cup \text{GEN}_{\text{NF}}(\mathcal{R}'), \mathbf{m})$, *if*
  - f *is* $\mathbf{m}$,
  - $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi} = \mathcal{P}$,
  - $\mathcal{R}_{\succsim_\pi} \supseteq \mathcal{U}^\pi_{\mathcal{R}}(\mathcal{P})$,
  - $\mathsf{gen} \to s \in \mathcal{U}^\pi_{\mathcal{R}}(\mathcal{P})$ *(for any term s) implies* $\mathcal{G}en_{\text{NF}}^{\succsim^\pi}$, *and*
  - $\succsim$ *is $\mathcal{C}_{\mathcal{E}}$-compatible;*[7]
- $(t_0, \mathcal{P} \setminus \mathcal{P}_{\succ_\pi}, \mathcal{R} \cup \text{GEN}_{\text{NF}}(\mathcal{R}'), \mathbf{a})$, *if*
  - f *is* $\mathbf{a}$,
  - $\mathcal{P}_{\succ_\pi} \cup \mathcal{P}_{\succsim_\pi} = \mathcal{P}$, *and*
  - $\mathsf{gen} \to s \in \mathcal{GU}^\pi(t_0, \mathcal{P}, \mathcal{R}, \pi)$ *(for any term s) implies* $\mathcal{G}en_{\text{NF}}^{\succsim^\pi}$, *and*
  - $\mathcal{R}_{\succsim_\pi} \supseteq \mathcal{GU}^\pi(t_0, \mathcal{P}, \mathcal{R}, \pi)$;
- $(t_0, \mathcal{P}, \mathcal{R} \cup \text{GEN}_{\text{NF}}(\mathcal{R}'), f)$, *otherwise.*

where $\mathcal{G}en_{\text{NF}}^{\succsim^\pi}$ is the finite set of constraints defined by

$$\mathcal{G}en_{\text{NF}}^{\succsim^\pi} = \{\mathsf{gen} \succsim_\pi \mathtt{f}(\overbrace{\mathsf{gen}, \ldots, \mathsf{gen}}^{n \text{ times}})\} \mid \mathtt{f}/n \in \mathit{Partial}(\mathcal{R}')\}$$

and $\mathit{Partial}(\mathcal{R}) \subset \mathcal{D}$ is the set

$$\mathit{Partial}(\mathcal{R}) = \mathcal{D} \setminus \{\mathtt{f} \mid \mathtt{f}(\overbrace{x_1, \ldots, x_n}^{\mathit{distinct}}) \to r \in \mathcal{R}\}$$

Proc is a sound and complete GDP processor.

Proof. This theorem is a particular case of Theorem 5.21, and therefore we focus on the differences. Namely, the replacement of the constraints $(\text{GEN}_{\text{NF}}(\mathcal{R}'))_{\succsim_\pi}$ corresponding to the infinite set of rules $\text{GEN}_{\text{NF}}(\mathcal{R}')$, by the finite set $\mathcal{G}en_{\text{NF}}^{\succsim^\pi}$. To prove that

[7]A quasi-rewrite order $\succsim$ is $\mathcal{C}_{\mathcal{E}}$-*compatible* if for a new function symbol c, $\mathsf{c}(x, y) \succsim x$ and $\mathsf{c}(x, y) \succsim y$.

this change is sound, we show that every constraint in $(\textsc{gen}_{\textsc{nf}}(\mathcal{R}'))_{\succsim_\pi}$ is implied by $\mathcal{G}en_{\textsc{nf}}^{\succsim_\pi} \cup \mathcal{R}_{\succsim_\pi}$.

By definition every rule in $\textsc{gen}_{\textsc{nf}}(\mathcal{R}')$ is of the form $\mathsf{gen} \to t$, for $t \in \mathcal{T}(\Sigma) \cap (\textsc{nf}(\mathcal{R}) \setminus \mathcal{T}(\mathcal{C},\mathcal{V}))$. Then the left-hand side of every constraint is always gen. We proceed now by structural induction on the term in right-hand side of the constraint.

For the base case, let t be a 0-ary function $\mathtt{d} \in \mathcal{D}$. The induced constraint $\mathsf{gen} \succsim_\pi \mathtt{d}$ is trivially satisfied since it belongs directly to $\mathcal{G}en_{\textsc{nf}}^{\succsim_\pi}$ by definition.

For the inductive case, let $t = \mathtt{f}(t_1,\ldots,t_n)$ and we distinguish two cases:

- $\mathtt{f} \in \mathcal{C}$. By assumption there is a rule $\mathsf{gen} \to s$ which is usable, so it follows that all the rules in $\textsc{gen}_{C}(\mathcal{R})$ are usable, since they are of the form $\mathsf{gen} \to s'$. Therefore $(\textsc{gen}_{C}(\mathcal{R}))_{\succsim_\pi} \subseteq \mathcal{R}_{\succsim_\pi}$. By definition there is a rule $\mathsf{gen} \to \mathtt{f}(\mathsf{gen},\ldots,\mathsf{gen}) \in \textsc{gen}_{C}(\mathcal{R})$, and hence the corresponding constraint $\mathsf{gen} \succsim_\pi \mathtt{f}(\mathsf{gen},\ldots,\mathsf{gen})$ holds. By the induction assumption $\mathsf{gen} \succsim_\pi t_i$ for $1 \le i \le n$, and by monotonicity $\mathsf{gen} \succsim_\pi \mathtt{f}(t_1,\ldots,t_n)$ follows.

- $\mathtt{f} \in \mathcal{D}$. Then $\mathtt{f} \notin \mathit{Partial}(\mathcal{R})$, or otherwise $\mathtt{f}(t_1,\ldots,t_n)$ would not be a normal form. We have a constraint $\mathsf{gen} \succsim_\pi \mathtt{f}(\mathsf{gen},\ldots,\mathsf{gen}) \in \textsc{gen}_{\textsc{nf}}^{\succsim_\pi}$. By the induction assumption $\mathsf{gen} \succsim_\pi t_i$ for $1 \le i \le n$, and by monotonicity $\mathsf{gen} \succsim_\pi \mathtt{f}(t_1,\ldots,t_n)$ follows. □

Example 5.59. *Let $\mathcal{R}$ below be $\mathcal{R}_{inc}$ from Example 5.4, and consider termination of $\rightsquigarrow_{\mathcal{R}}$ derivations starting from the abstract goal $t^\alpha = \mathsf{p_{in}}(\mathsf{g})$. According to Theorem 5.54, narrowing terminates in $\mathcal{R}$ from the above goal if the GDP problem $(goal(x), \mathcal{P}', \mathcal{R}', \mathbf{a})$ is finite; where $\mathcal{R}'$ is $\widehat{\mathcal{R}_{goal}} \cup \textsc{gen}(\mathcal{R})$. Therefore we have the rules of $\mathcal{R}$ with a rule for the abstract goal, extra variables replaced by generators, the constructor generators, and the infinite set of root-defined normal form generators (which are omitted below):*

$$\begin{aligned}
\mathsf{p_{in}}(\mathsf{g}(X)) &\to \mathsf{u_3}(\mathsf{p_{in}}(X), X)\\
\mathsf{p_{in}}(X) &\to \mathsf{u_1}(\mathsf{q_{in}}(\mathsf{f}(\mathsf{gen})), X)\\
\mathsf{q_{in}}(\mathsf{g}(Y)) &\to \mathsf{q_{out}}(\mathsf{g}(Y))\\
\mathsf{u_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) &\to \mathsf{u_2}(\mathsf{p_{in}}(Y), X, Y)\\
\mathsf{u_2}(\mathsf{p_{out}}(Y), X, Y) &\to \mathsf{p_{out}}(X)\\
\mathsf{u_3}(\mathsf{p_{out}}(X), X) &\to \mathsf{p_{out}}(\mathsf{g}(X))\\
\mathsf{goal}(x) &\to \mathsf{p_{in}}(x)\\
\mathsf{gen} &\to \mathsf{f}(\mathsf{gen})\\
\mathsf{gen} &\to \mathsf{g}(\mathsf{gen})\\
\mathsf{gen} &\to \mathsf{p_{out}}(\mathsf{gen})\\
\mathsf{gen} &\to \mathsf{q_{out}}(\mathsf{gen})
\end{aligned}$$

$\mathcal{P}'$ is simply $\widehat{DP(\mathcal{R}_{goal})}$, i.e. the dependency pairs of $\mathcal{R}_{goal}$, with extra variables

replaced by generators:

$$\mathsf{P_{in}}(\mathsf{g}(X)) \to \mathsf{P_{in}}(X) \tag{5.10}$$

$$\mathsf{P_{in}}(\mathsf{g}(X)) \to \mathsf{U_3}(\mathsf{p_{in}}(X), X) \tag{5.11}$$

$$\mathsf{P_{in}}(X) \to \mathsf{Q_{in}}(\mathsf{f}(\mathsf{gen})) \tag{5.12}$$

$$\mathsf{P_{in}}(X) \to \mathsf{U_1}(\mathsf{q_{in}}(\mathsf{f}(\mathsf{gen})), X) \tag{5.13}$$

$$\mathsf{U_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) \to \mathsf{U_2}(\mathsf{p_{in}}(Y), X, Y) \tag{5.14}$$

$$\mathsf{U_1}(\mathsf{q_{out}}(\mathsf{f}(Y)), X) \to \mathsf{P_{in}}(Y) \tag{5.15}$$

$$\mathsf{GOAL}(x) \to \mathsf{P_{in}}(x) \tag{5.16}$$

Note that the only initial pair is (5.16)*. The estimation of the dependency graph is shown below, where elements not belonging to the goal–directed dependency graph are greyed out.*

$$\begin{tikzcd}
(5.15) \arrow[d, dotted] \arrow[dr, dotted] \arrow[drr, dotted] \arrow[ddr, dotted] & (5.14) & (5.16) \arrow[d] \arrow[dl] \arrow[dll] \arrow[ddl] \\
(5.13) & (5.12) & (5.11) \\
& (5.10) \arrow[ul] \arrow[u] \arrow[ur] \arrow[loop below] &
\end{tikzcd}$$

The only SCC in the directed–goal dependency graph is the pair $\mathsf{P_{in}}(\mathsf{g}(X)) \to \mathsf{P_{in}}(X)$*, and that there is only the trivial path from the initial pair* (5.16) *to this pair, which includes both. In this case the goal-directed usable rules are trivially the empty set, and hence the set* $\mathcal{G}en^{\succsim\pi}_{\mathrm{NF}}$ *of additional constraints need not be considered (even if* $\mathcal{R}_{goal}$ *is not left-linear). From here it is easy to find an RPO that orients the pair and solves the termination problem. On the other hand, the technique of [Schneider-Kamp et al., 2009b] needs a global argument filtering that removes every extra variable, which ultimately has to filter either the argument of* $\mathsf{P_{in}}$ *or* g*, precluding a successful termination proof. The same remark applies to [Nishida and Vidal, 2010; Vidal, 2008] after filtering out the extra variables.*

In conclusion, we have proven in this section that with a modest extension, the approach can be adapted to automatically prove termination of narrowing in non left-linear constructor systems. As we have seen, in practice it is not necessary to regard the infinite set $\textsc{gen}_{\mathrm{NF}}(\mathcal{R})$ of root-defined normal forms generators, and instead it is possible to use the reduction pairs processor of Theorem 5.58 which imposes additional constraints instead.

5.3.3 Admissible Derivations

The termination criterion of Theorem 5.54 based on relative termination of rewriting with generators is not complete. Given a TRS $\mathcal{R}$ and an abstract goal t_α, it can be that every $\rightsquigarrow_{\mathcal{R}}$ derivation stemming from a concretization of t_α terminates while the GDP problem $(\mathsf{goal}(x_1,\ldots,x_n), DP(\widehat{\mathcal{R}_{t^\alpha}}), \widehat{\mathcal{R}_{t^\alpha}} \cup \textsc{gen}(\mathcal{R}), \mathbf{a})$ is not finite. This is caused by the loss of sharing: repeated variables in non linear terms of the derivation are replaced by independent occurrences of gen, which can give rise to different values. As a result, not every $\rightarrow_{\textsc{gen}(\mathcal{R})}$ derivation has a corresponding $\rightsquigarrow_{\mathcal{R}}$ derivation.

Example 5.60. *Consider the TRS $\mathcal{R}$ given by the single rule $\mathtt{f}(\mathsf{a},\mathsf{b},x) \rightarrow \mathtt{f}(x,x,x)$. Narrowing terminates in $\mathcal{R}$ for any goal, as it can be proven in the framework of Chapter 4: the corresponding NDP problem is given by $\mathcal{R}$ and a single dependency pair $\mathtt{F}(\mathsf{a},\mathsf{b},x) \rightarrow \mathtt{F}(x,x,x)$. Then a single application of the Dependency Pair processor of Theorem 4.41 can show the absence of cycles*[8]*.*

However, both the approach of this chapter and [Nishida and Vidal, 2010; Vidal, 2008] fail to prove termination of the abstract goal $\mathtt{f}(\mathbf{v},\mathbf{g},\mathbf{g})$. Following our approach, one applies Theorem 5.54 and obtains the GDP problem given by the goal $t_0 = \mathtt{goal}(x,y)$, the rules:

$$\begin{aligned} \mathtt{f}(\mathsf{a},\mathsf{b},x) &\rightarrow \mathtt{f}(x,x,x) \\ \mathsf{gen} &\rightarrow \mathsf{a} \\ \mathsf{gen} &\rightarrow \mathsf{b} \\ \mathtt{goal}(x,y) &\rightarrow \mathtt{f}(\mathsf{gen},x,y) \end{aligned}$$

and the pair:

$$\mathtt{F}(\mathsf{a},\mathsf{b},x) \rightarrow \mathtt{F}(x,x,x) \tag{5.17}$$

which is not relatively terminating, as witnessed by the infinite $\rightarrow_{\mathcal{R}} / \rightarrow_{\textsc{gen}(\mathcal{R})}$ derivation:

$$\mathtt{f}(\underline{\mathsf{gen}},\underline{\mathsf{gen}},\mathsf{gen}) \rightarrow^*_{\textsc{gen}(\mathcal{R})} \underline{\mathtt{f}(\mathsf{a},\mathsf{b},\mathsf{gen})} \rightarrow_{\mathcal{R}} \underline{\mathtt{f}(\mathsf{gen},\mathsf{gen},\mathsf{gen})} \rightarrow^*_{\textsc{gen}(\mathcal{R})} \cdots$$

This issue was already pointed out by [*Antoy and Hanus*, 2006] and further discussed in [*Vidal*, 2008; *Nishida and Vidal*, 2010]. It can be modelled using the notion of *admissible* derivations.

Definition 5.61 (Admissible derivation [*Antoy and Hanus*, 2006])**.** *Let* $\textsc{gen}(\mathcal{R})$ *be a TRS over* $\Sigma \cup \{\mathsf{gen}\}$ *and* $t \in \mathcal{T}(\Sigma,\mathcal{V})$ *be a term. A derivation* $\mathcal{D}$ *is called admissible if all the occurrences of* gen *originating from the instantiation of the same variable are reduced to the same term.*

The reader can check that the infinite derivation of Example 5.60 is not admissible.

A formalism for ensuring that only admissible derivations are possible, based on a decomposition of terms in skeleton and enviroment, can be found in [*Antoy and Hanus*, 2006]. By means of this notion, one can give a complete version of Theorem 5.52 which considers only admissible derivations. This is done by [*Nishida and Vidal*, 2010] (Lemma 2 and Theorem 2).

[8]The naive estimation used in Chapter 4 cannot actually prove absence of cycles in this example, but a more advanced estimation such as the EDG* of [*Middeldorp*, 2002] can.

Although a very desirable extension, the termination of admissible derivations is beyond the scope of this thesis and planned for future work. We only suggest that the approach presented in this chapter is complemented with the dependency graph processor of Theorem 4.41, which operates directly on narrowing derivations and does not suffer from the uncompleteness problem.

5.4 Termination of Logic Programs

In this section, we briefly consider the application of these results to prove the termination of logic programs in terms of the termination of narrowing.

As in the case of standard rewriting, in TRSs without extra variables there is a lifting between infinitary constructor rewriting derivations and narrowing derivations with possibly infinite terms. But in the standard rewriting case we saw, in Proposition 2.2, that the lifting is lost in presence of extra variables. The constructor restriction brings the lifting back, as formalized by the following proposition.

Proposition 5.62 (Extended Lifting Lemma for Infinitary Constructor Rewriting)**.** *Let $\mathcal{R}$ be a generalized TRS over a signature Σ. For all term $t \in \mathcal{T}^\infty(\Sigma, \mathcal{V})$ and constructor substitution $\theta : \mathcal{V} \to \mathcal{T}^\infty(\mathcal{C}, \mathcal{V})$, if $s\theta \overset{\infty}{\to}{}^*_{\mathcal{R}} t$ then there exist a term s' and substitutions θ' and σ such that:*

- $s \rightsquigarrow^*_{\sigma,\mathcal{R}} s'$,
- $s'\theta' = t$,
- $\sigma\theta' = \theta\ [\mathcal{V}ar(s)]$,
- $\theta' : \mathcal{V} \to \mathcal{T}^\infty(\mathcal{C}, \mathcal{V})$ *is a constructor substitution.*

and one can assume that the narrowing derivation uses the same rules at the same positions as the rewriting derivation.

Proof. We offer only an informal proof, which proceeds by induction on the length of the rewriting derivation. The base case is trivial. For the inductive case, consider the rewriting sequence $s\theta \overset{\infty}{\to}{}^p_{l\to r} t \overset{\infty}{\to}{}^*_{\mathcal{R}} t'$. Then there is a substitution σ such that $(s\theta) = l\sigma$ and the substitution $\tau = \sigma\theta$ is a unifier for s and l. Notice that σ must be a constructor substitution because of the constructor rewriting restriction. Hence τ is a constructor substitution by construction. Moreover, as the domains are disjoint, $\sigma\theta = \sigma \cup \theta$. Therefore there is a most general unifier[9] τ' of s and l, and there is a substitution ρ such that $\tau = \tau'\rho$. Again, τ' and ρ are constructor substitutions. As θ is normalized, then $p \in \mathcal{P}os(s)$ and $(s\theta)|_p = s|_p\theta$. By definition, $s \overset{p}{\rightsquigarrow}_{\tau',\mathcal{R}} s'$. Applying the equivalences obtained so far, we obtain $s'\rho = s\tau'\rho[r\tau'\rho]_p = s\theta[r\sigma]_p = t$. As $\rho\restriction_{s'}$ is constructor and therefore normalized, the induction hypothesis yields $s' \rightsquigarrow^*_{\tau'',\mathcal{R}} s''$ and a constructor substitution ρ' such that $\tau''\rho' = \rho\ [\mathcal{V}ar(s'')]$. Hence $s \rightsquigarrow^*_{\tau'\tau'',\mathcal{R}} s''$, $\theta' = \rho\rho'$ is a constructor substitution, and $s''\theta' = t'$. $\square$

[9]And as usual, idempotent by assumption

Roughly speaking, the finiteness of all $\rightsquigarrow_{\mathcal{R}_{\mathcal{P}}}$ derivations starting from a set of terms $p_{in}(\ldots)$ entails the termination of the query $p(\ldots)$ over the logic program $\mathcal{P}$. The next proposition formalizes this. In the following, argument filterings are used as a convenient tool to characterize the set of queries considered, and $\bar{t}$ denotes a sequence of terms $t_1, \ldots, t_n$.

Proposition 5.63 (Termination of logic programs via termination of narrowing)**.** *Let Δ be a set of predicate symbols, $\mathcal{P}$ be a logic program and π be a non-collapsing argument filter over $\Sigma \cup \Delta$, extended so that $\pi(p_{in}) = \pi(p)$ for all $p \in \Delta$. Define the set of terms $S = \{p_{in}(\bar{t}) \mid p \in \Delta, t_i \in \mathcal{T}(\Sigma, \mathcal{V}), \pi(p_{in}(\bar{t}) \in \mathcal{T}(\Sigma_\pi)\}$.*

If all terms $s \in S$ are $(\rightsquigarrow_{\mathcal{R}_{\mathcal{P}}})$-terminating, then all logic queries $Q \in \mathcal{A}(\Sigma, \Delta, \mathcal{V})$ with $\pi(Q) \in \mathcal{A}(\Sigma_\pi, \Delta_\pi)$ are terminating in $\mathcal{P}$.

We need to introduce two auxiliary results. The first is a mapping between logic and narrowing derivations.

Lemma 5.64. *Let $\mathcal{P}$ be a logic program, $\bar{t}$ be terms from $\mathcal{T}(\Sigma, \mathcal{V})$, and let $p(\bar{t}) \vdash_{\mathcal{P},\sigma} Q$. If $Q = \Box$ then $p_{in}(\bar{t}) \rightsquigarrow^*_{\sigma,\mathcal{R}_P} p_{out}(\bar{t})\sigma$. Otherwise, if Q is $q(\overline{v})$, then $p_{in}(\bar{t}) \rightsquigarrow^*_{\sigma,\mathcal{R}_P} r \trianglerighteq q_{in}(\overline{v})$.*

Proof. By [Schneider-Kamp et al., 2009b, Lemma 3.4] there is a constructor rewriting derivation $p_{in}(\bar{t})\sigma \rightarrow^*_{cR_P} p_{out}(\bar{t})\sigma$ if $Q = \Box$, or $p_{in}(\bar{t})\sigma \rightarrow^*_{\mathcal{R}_P} r \trianglerighteq q_{in}(\overline{v})$ if $Q = q(\overline{v})$, and in both cases σ is constructor. The lifting lemma for constructor rewriting with extra variables of Proposition 5.62 yields $p_{in}(\bar{t}) \rightsquigarrow^*_{\sigma',\mathcal{R}_P} p_{out}(\bar{t})\sigma'$ and $p_{in}(\bar{t}) \rightsquigarrow^*_{\sigma',\mathcal{R}_P} r'$. respectively. Since the original substitution σ computed by the logic program is a composition of mgu's, it can be shown that $\sigma' = \sigma$. Hence $r' = r \trianglerighteq q_{in}(\overline{v})$, completing the proof. ☐

We also need another lemma that allows us to focus on single-atom non-terminating queries.

Lemma 5.65 (Non-terminating queries [Schneider-Kamp et al., 2009b, Lemma 3.5])**.** *Let $\mathcal{P}$ be a logic program. For all infinite derivation $Q_0 \vdash_{\mathcal{P}} Q_1 \vdash_{\mathcal{P}} \cdots$, there is a Q_i of the form $q(\overline{v}), \ldots$ with $i > 0$ such that the query $q(\overline{v})$ is also non-terminating.*

We can now proceed with the proof of Proposition 5.63

Proof of Proposition 5.63. Follows from the proof of [Schneider-Kamp et al., 2009b, Theorem 3.7], by making use of Lemma 5.64 in places where the analog lemma for infinitary rewriting was used.

Assume that there is a non-terminating query $p(\bar{t})$ as above, with $p(\bar{t}) \vdash_{\mathcal{P}} Q_1 \vdash Q_2 \cdots$. By Lemma 5.65 there is an $i_1 > 0$ with $Q_{i_1} = q_1(\overline{v_1})$, and an infinite derivation $q_1(\overline{v_1}) \vdash_{\mathcal{P}} Q'_1 \vdash_{\mathcal{P}} Q'_2 \vdash_{\mathcal{P}} \cdots$. Lemma 5.64 yields $p_{in}(\bar{t}) \rightsquigarrow^*_{\mathcal{R}_{\mathcal{P}},\sigma} r \trianglerighteq q_{1_in}(\overline{v_1})$. Applying Lemma 5.65 again, there is an $i_2 > 0$ with $Q'_{i_2} = q_2(\overline{v_2}), \ldots$, and an infinite derivation $q_2(\overline{v_2}) \vdash_{\mathcal{P}} Q''_1 \vdash_{\mathcal{P}} Q''_2 \vdash_{\mathcal{P}} \cdots$. Lemma 5.64 yields $r_1 \rightsquigarrow^*_{\mathcal{R}_{\mathcal{P}},\sigma} r_2 \trianglerighteq q_{2_in}(\overline{v_2})$. By this reasoning, we obtain an infinite narrowing sequence starting from $p_{in}(\bar{t})$ But since $\pi(p(\bar{t})) \in \mathcal{A}(\Sigma_\pi, \Delta_\pi)$, and hence $\pi(p_{in}(\bar{t})) \in \mathcal{T}(\Sigma_\pi)$, which is a contradiction. ☐

Recall the logic program $\mathcal{P}_{inc}$, and the resulting TRS $\mathcal{R}_{inc}$, of Example 5.4. Proving $(\rightsquigarrow_{\mathcal{R}_{inc}})$-termination of the abstract goal $\mathtt{p}(\mathbf{g})$, as it is done in Example 5.59, implies that every query $\mathtt{p}(t)$ for a ground term t terminates in $\mathcal{P}_{inc}$. Similarly, proving $(\rightsquigarrow_{\mathcal{R}_{inc}})$-termination of the abstract goal $\mathtt{p}(\mathbf{v})$[10] would imply that every query $\mathtt{p}(t)$ for a term, possibly with variables, t terminates in $\mathcal{P}_{inc}$.

Since we prefer to focus on the novel aspects of termination of narrowing in this thesis, we conclude this brief section by referring the interested reader to [*Schneider-Kamp et al.*, 2009b; *Schneider-Kamp*, 2008].

5.5 Results and Discussion

The technique for proving termination of narrowing introduced in this chapter is strictly more general than [*Nishida and Vidal*, 2010; *Vidal*, 2008], which is restricted to left-linear systems with no extra variables. Our technique is also much more efficient in practice, thanks to the replacement of search heuristics for the global argument filtering by a constraint guided approach. A considerable shortcoming of our approach is the loss of minimality. This means that many desirable techniques, such as the subterm criterion of [*Hirokawa and Middeldorp*, 2004], cannot be applied without restrictions. The same remarks apply when comparing our new approach to the infinitary rewriting framework of [*Schneider-Kamp et al.*, 2009b], which also employs a global argument filtering and heuristics. Even so, we expect that the ability to automatically infer the best argument filtering without the use of heuristics makes up for this shortcoming.

In order to see how well the new approach behaves in practice, we benchmark it versus the termination of narrowing method of [*Vidal*, 2008; *Nishida and Vidal*, 2010] and the termination of logic programs method of [*Schneider-Kamp et al.*, 2009b].

We employ two sets of examples generated from the Logic Programming category of the Termination Problem Database (TPDB) 5.0 [*Marche and Zantema*, 2007]. The first set, called LPclean here, turns these logic programs into TRSs using the transformation of Definition 5.2. After excluding those examples containing meta cuts, arithmetic or other non declarative primitives, there are in total 295 termination problems.

The second set, called LPsolve here, uses the same 295 logic programs as base. Each logic program is transformed into a new "*meta*" logic program, and then transformed into a TRS using again the transformation of Definition 5.2. For example, given the logic program for the ackerman function:

```
ack(0,N,s(N)).
ack(s(M),0,Res) :- ack(M,s(0),Res).
ack(s(M),s(N),Res) :- ack(s(M),N,Res1),ack(M,Res1,Res).
```

the "meta" version generated by this program is:

[10]This can be shown using the star-estimation of the dependency graph of [*Hirokawa and Middeldorp*, 2004].

```
clause(ack(0,N,s(N)),true).
clause(ack(s(M),0,Res),ack(M,s(0),Res)).
clause(ack(s(M),s(N),Res),(ack(s(M),N,Res1),ack(M,Res1,Res))).
solve(true).
solve((X,Y)) :- solve(X),solve(Y).
solve(X) :- clause(X,Y),solve(Y).
goal(A,B,C) :- solve(ack(A,B,C)).
```

This transformation preserves termination, while at the same time heavily altering the shape of the program, arguably making the job of termination provers much harder. The reason for including the LPsolve sets of problems is to observe how heuristic based approaches are highly dependent on the shape of the termination problem. We argue that there is no set of heuristics that works for all cases, and it is doubtful that an automated tool can detect which heuristic is most appropriate for a particular problem in all cases. In contrast, our approach is not affected by this since it is fully constraint based.

In both example sets, the resulting TRSs are non-left-linear and contain extra variables, and hence [*Nishida and Vidal*, 2010; *Vidal*, 2008] cannot prove termination of narrowing. Even so, for the sake of the benchmark, we filter extra variables from the initial problem and proceed to apply their method to the filtered system. Although the resulting TRS is still non-left-linear and hence the method cannot be used to prove termination of narrowing, we still find the results can be useful for orientative purposes.

A huge number of techniques for termination have been developed in the recent years. In order to provide a fair comparison we focus on a small set of DP processors: the dependency graph processor, the RPO reduction pair implemented by means of the SAT encoding of [*Codish et al.*, 2005; *Schneider-Kamp et al.*, 2007; *Codish et al.*, 2006] extended to account for our notion of goal-directed usable rules, the subterm criterion of [*Hirokawa and Middeldorp*, 2004] and the narrowing and instantiation graph refinement processors [*Arts and Giesl*, 2000]. It is our opinion that this set of processors constitutes a realistic tool which can be used to solve a very reasonable class of termination problems. The (very fast) subterm criterion processor is included to illustrate the shortcomings of the loss of minimality.

We have implemented our approach in the termination tool narradar. narradar recognizes the TPDB format [*Marche and Zantema*, 2007] with extensions for expressing initial goals and narrowing. In order to perform the test we also implemented the approach of [*Nishida and Vidal*, 2010; *Vidal*, 2008] and [*Schneider-Kamp et al.*, 2009b].

- For [*Nishida and Vidal*, 2010; *Vidal*, 2008] narradar uses three different heuristics for computing argument filters:

  heu is the binding-time analysis described in [*Nishida and Vidal*, 2010].

  inn is a naive heuristic which always filters the innermost position.

  search generates and tries all the candidate filterings in depth-first search style.

LPclean	# Successes	Avg. success time
narradar	174	0.97 seconds
[*Schneider-Kamp et al.*, 2009b]	168	0.23 seconds
[*Vidal*, 2008] inn	157	0.30 seconds
[*Vidal*, 2008] search	151	2.46 seconds
[*Nishida and Vidal*, 2010] heu	120	0.17 seconds
LPsolve	**# Successes**	**Avg. success time**
narradar	89	0.79 seconds
[*Vidal*, 2008] inn	84	0.40 seconds
[*Vidal*, 2008] search	64	13.71 seconds
[*Schneider-Kamp et al.*, 2009b]	56	0.27 seconds
[*Nishida and Vidal*, 2010] heu	0	

Table 5.1: Benchmark results

- For [*Schneider-Kamp et al.*, 2009b] narradar uses the unbounded positions heuristic which does a type analysis of the logic program and computes an optimal filtering. Note that while [*Schneider-Kamp et al.*, 2009b] introduces two type-based heuristics, we consider only the most powerful one.

All the problems were run in automated mode on a 2.5Ghz Intel CPU with a 60 seconds timeout The results are displayed in Table 5.1.

The new approach is the best performer in the LPclean set of examples, beating [*Schneider-Kamp et al.*, 2009b] by 6 examples and the best performing heuristic of [*Vidal*, 2008] by 17 examples. The time to produce a proof is 0.74 seconds higher on average, mainly because the fast subterm criterion processor cannot be applied, but also because the constraints to solve are larger. We observe that the binding-time analysis heuristic of [*Nishida and Vidal*, 2010] performs rather poorly; the reason is that this heuristic *never* filters constructor symbols, and therefore seems to be of limited usefulness in practice.

The LPsolve set of examples draws a more extreme picture. Our approach beats the previous best contender, [*Schneider-Kamp et al.*, 2009b], by a total of 33 examples, solving 45% more termination problems. As expected, the type heuristic used by [*Schneider-Kamp et al.*, 2009b] is easily confused by the program transformation, while our approach is not affected since it does not rely on a heuristic. For the programs in the LPsolve set, an innermost heuristic often produces the best results, as can be seen from the results of [*Vidal*, 2008] *inn*. On the other hand, since the predicates of the original program become constructor symbols in the transformed program, filtering on constructor symbols is essential here; correspondingly, the binding-time analysis of [*Vidal*, 2008] does not manage to solve any example in the LPsolve set.

Let us come back to the LPclean results. The new approach solves a total of 9 examples that none of the previous approaches manage to solve using solely the set of processors mentioned above:

```
BCGGV05/map_color.pl
```

```
BCGGV05/transpose-bf.pl
SGST06/incomplete.pl
talp/dds/reverse-iio.pl
talp/plumer/pl8.4.2.pl
talp/talp/binary2.pl
terminweb/cti/som.pl
terminweb/type-based/transpose.pl
```

The unsoundness of Theorem 5.54 related to the issue of admissible derivations commented in Section 5.3.3 is responsible for one missed example (`SGST06/toyama.pl`, developed above in Example 5.60) with regard to the approach of [*Schneider-Kamp et al.*, 2009b]. This example can be solved by combining the approach of this chapter with the dependency graph of Chapter 4 (cf. section 5.3.3).

Aside from `SGST06/toyama.pl`, there are two examples remaining which our technique does not handle, whereas the previously existing approaches do:

```
SGST06/intlist.pl
SGST06/prime.pl
```

We consider the case of `SGST06/intlist.pl` below.

Example 5.66. *Consider the TRS $\mathcal{R}$ below, obtained after transforming* `SGST06/intlist.pl` *from the logic programming category of the Termination Problem Database into a TRS.*

$$\begin{aligned}
\mathtt{int_{in}}(\mathtt{0},\mathtt{0},\mathtt{cons}(\mathtt{0},\mathtt{nil})) &\to \mathtt{int_{out}}(\mathtt{0},\mathtt{0},\mathtt{cons}(\mathtt{0},\mathtt{nil}))\\
\mathtt{int_{in}}(\mathtt{0},\mathtt{s}(y),\mathtt{cons}(\mathtt{0},xs)) &\to \mathtt{u_2}(\mathtt{int_{in}}(\mathtt{s}(\mathtt{0}),\mathtt{s}(y),xs),xs,y)\\
\mathtt{int_{in}}(\mathtt{s}(x),\mathtt{0},nil) &\to \mathtt{int_{out}}(\mathtt{s}(x),\mathtt{0},nil)\\
\mathtt{int_{in}}(\mathtt{s}(x),\mathtt{s}(y),xs) &\to \mathtt{u_3}(\mathtt{int_{in}}y(x,y,zs),x,xs,y)\\
\mathtt{intlist_{in}}(\mathtt{cons}(x,xs),\mathtt{cons}(\mathtt{s}(x),ys)) &\to \mathtt{u_1}(\mathtt{intlist_{in}}(xs,ys),x,xs,ys)\\
\mathtt{u_1}(\mathtt{intlist_{out}}(xs,ys),x,xs,ys) &\to \mathtt{intlist_{out}}(\mathtt{cons}(x,xs),\mathtt{cons}(\mathtt{s}(x),ys))\\
\mathtt{u_2}(\mathtt{int_{out}}(\mathtt{s}(\mathtt{0}),\mathtt{s}(y),xs),xs,y) &\to \mathtt{int_{out}}(\mathtt{0},\mathtt{s}(y),\mathtt{cons}(\mathtt{0},xs))\\
\mathtt{u_3}(\mathtt{int_{out}}(x,y,zs),x,xs,y) &\to \mathtt{u_4}(\mathtt{intlist_{in}}(zs,xs),x,xs,y,zs)\\
\mathtt{u_4}(\mathtt{intlist_{out}}(zs,xs),x,xs,y,zs) &\to \mathtt{int_{out}}(\mathtt{s}(x),\mathtt{s}(y),xs)
\end{aligned}$$

Narrowing derivations in this TRS starting from the abstract goal $t^\alpha = \mathtt{int_{in}}(\mathbf{g},\mathbf{g},\mathbf{v})$ are finite as far as we can tell. Following Theorem 5.54, narrowing terminates in $\mathcal{R}$ from t^α if the GDP problem $(goal(x),\mathcal{P}',\mathcal{R}',\mathbf{a})$ is finite; where $\mathcal{R}'$ is $\widehat{\mathcal{R}_{goal}} \cup \mathrm{GEN}(\mathcal{R})$, that is, $\mathcal{R}$ plus a rule for the abstract goal, extra variables replaced by generators, together with the generator rules (we omit the rules for root-defined normal forms):

$$\mathtt{int_{in}}(\mathtt{0},\mathtt{0},\mathtt{cons}(\mathtt{0},\mathtt{nil})) \to \mathtt{int_{out}}(\mathtt{0},\mathtt{0},\mathtt{cons}(\mathtt{0},\mathtt{nil})) \tag{5.18}$$

$$\mathtt{int_{in}}(\mathtt{0},\mathtt{s}(y),\mathtt{c}(\mathtt{0},xs)) \to \mathtt{u_2}(\mathtt{int_{in}}(\mathtt{s}(\mathtt{0}),\mathtt{s}(y),xs),xs,y) \tag{5.19}$$

$$\mathtt{int_{in}}(\mathtt{s}(x),\mathtt{0},nil) \to \mathtt{int_{out}}(\mathtt{s}(x),\mathtt{0},nil) \tag{5.20}$$

$$\begin{tikzcd}
 & & (5.35) & & & \\
(5.42) \arrow[urr, dotted] \arrow[r] \arrow[dr] \arrow[drr, bend right] & (5.36) \arrow[loop above] \arrow[ur, dotted] \arrow[d, bend left] \arrow[dr] & (5.41) & & (5.38) \arrow[loop above] \arrow[dr] & \\
 & (5.34) \arrow[u, bend left] \arrow[r] & (5.37) \arrow[u] \arrow[r] & (5.40) \arrow[ur] \arrow[rr] & & (5.39)
\end{tikzcd}$$

Figure 5.3: Dependency Graph of Example 5.66

$$\mathtt{int_{in}}(\mathtt{s}(x), \mathtt{s}(y), xs) \to \mathtt{u_3}(\mathtt{int_{in}}(x, y, \mathsf{gen}), x, xs, y) \tag{5.21}$$
$$\mathtt{intlist_{in}}(\mathtt{c}(x, xs), \mathtt{c}(\mathtt{s}(x), ys)) \to \mathtt{u_1}(\mathtt{intlist_{in}}(xs, ys), x, xs, ys) \tag{5.22}$$
$$\mathtt{u_1}(\mathtt{intlist_{out}}(xs, ys), x, xs, ys) \to \mathtt{intlist_{out}}(\mathtt{c}(x, xs), \mathtt{c}(\mathtt{s}(x), ys)) \tag{5.23}$$
$$\mathtt{u_2}(\mathtt{int_{out}}(\mathtt{s}(\mathtt{0}), \mathtt{s}(y), xs), xs, y) \to \mathtt{int_{out}}(\mathtt{0}, \mathtt{s}(y), \mathtt{c}(\mathtt{0}, xs)) \tag{5.24}$$
$$\mathtt{u_3}(\mathtt{int_{out}}(x, y, zs), x, xs, y) \to \mathtt{u_4}(\mathtt{intlist_{in}}(zs, xs), x, xs, y, zs) \tag{5.25}$$
$$\mathtt{u_4}(\mathtt{intlist_{out}}(zs, xs), x, xs, y, zs) \to \mathtt{int_{out}}(\mathtt{s}(x), \mathtt{s}(y), xs) \tag{5.26}$$
$$\mathtt{goal}(x, y) \to \mathtt{int_{in}}(x, y, \mathsf{gen}) \tag{5.27}$$
$$\mathsf{gen} \to \mathtt{0} \tag{5.28}$$
$$\mathsf{gen} \to \mathtt{s}(\mathsf{gen}) \tag{5.29}$$
$$\mathsf{gen} \to \mathtt{nil} \tag{5.30}$$
$$\mathsf{gen} \to \mathtt{c}(\mathsf{gen}, \mathsf{gen}) \tag{5.31}$$
$$\mathsf{gen} \to \mathtt{int_{out}}(\mathsf{gen}) \tag{5.32}$$
$$\mathsf{gen} \to \mathtt{intlist_{out}}(\mathsf{gen}) \tag{5.33}$$

The dependency pairs of the problem are listed below:

$$\mathtt{INT_{in}}(\mathtt{0}, \mathtt{s}(y), \mathtt{c}(\mathtt{0}, xs)) \to \mathtt{INT_{in}}(\mathtt{s}(\mathtt{0}), \mathtt{s}(y), xs) \tag{5.34}$$
$$\mathtt{INT_{in}}(\mathtt{0}, \mathtt{s}(y), \mathtt{c}(\mathtt{0}, xs)) \to \mathtt{U_2}(\mathtt{int_{in}}(\mathtt{s}(\mathtt{0}), \mathtt{s}(y), xs), xs, y) \tag{5.35}$$
$$\mathtt{INT_{in}}(\mathtt{s}(x), \mathtt{s}(y), xs) \to \mathtt{INT_{in}}(x, y, \mathsf{gen}) \tag{5.36}$$
$$\mathtt{INT_{in}}(\mathtt{s}(x), \mathtt{s}(y), xs) \to \mathtt{U_3}(\mathtt{int_{in}}(x, y, \mathsf{gen}), x, xs, y) \tag{5.37}$$
$$\mathtt{INTLIST_{in}}(\mathtt{c}(x, xs), \mathtt{c}(\mathtt{s}(x), ys)) \to \mathtt{INTLIST_{in}}(xs, ys) \tag{5.38}$$
$$\mathtt{INTLIST_{in}}(\mathtt{c}(x, xs), \mathtt{c}(\mathtt{s}(x), ys)) \to \mathtt{U_1}(\mathtt{intlist_{in}}(xs, ys), x, xs, ys) \tag{5.39}$$
$$\mathtt{U_3}(\mathtt{int_{out}}(x, y, zs), x, xs, y) \to \mathtt{INTLIST_{in}}(zs, xs) \tag{5.40}$$
$$\mathtt{U_3}(\mathtt{int_{out}}(x, y, zs), x, xs, y) \to \mathtt{U_4}(\mathtt{intlist_{in}}(zs, xs), x, xs, y, zs) \tag{5.41}$$
$$\mathtt{GOAL}(x, y) \to \mathtt{INT_{in}}(x, y, \mathsf{gen}) \tag{5.42}$$

where the single initial pair is 5.42. The corresponding dependency graph is shown in Figure 5.3. We have three SCCs: {5.38}, {5.36} and {5.34,5.36}.

Consider the problem given by the first SCC $\{5.38\}$. *To compute the usable rules we need to look at the dependency pairs in a path from an initial pair to this SCC, i,e,* $PATH_{\mathcal{G}_0}(\{5.42\}, \{5.38\})$, *which in this case is the set* $\{5.42, 5.34, 5.36, 5.37, 5.40, 5.38\}$. *Therefore we need a filtering which drops the generators in the right hand sides of pairs 5.42, 5.36 and 5.37, while enforcing the variable condition in the rest of the pairs in the path, as well as in the rules. We can start with* π_1, *defined as*

$$\begin{aligned}
\pi_1(\mathtt{INT_{in}}) &= [1,2]\\
\pi_1(\mathtt{INTLIST_{in}}) &= [1]\\
\pi_1(\mathtt{U_3}) &= [1,2,4]\\
\pi_1(\mathtt{int_{in}}) &= [1,2]\\
\pi_1(\mathtt{intlist_{in}}) &= [1]\\
\pi_1(\mathtt{u_1}) &= [1,2,3]\\
\pi_1(\mathtt{u_2}) &= [1,3]\\
\pi_1(\mathtt{u_3}) &= [1,2,4]\\
\pi_1(\mathtt{u_4}) &= [1,2,4,5]
\end{aligned}$$

and the identity for any other symbol. We use this filtering and the pairs in the path to compute the usable rules; in this case all the rules are usable (brought in by pair 5.37) except the rules of gen *and* goal. *In order to orient the rules,* NARRADAR *tries to compute a refinement* π_2 *of* π_1 *and an RPO reduction pair* $(\succ, \succeq)$ *such that* $l \succeq_\pi r$ *for each one of the usable rules. Unfortunately there is no such* π_2.

To see why, let us look at the filtered version of rule 5.19

$$\mathtt{int_{in}}(\mathtt{0}, \mathtt{s}(y)) \to \mathtt{u_2}(\mathtt{int_{in}}(\mathtt{s}(\mathtt{0}), \mathtt{s}(y)), y)$$

Note that the $\mathtt{int_{in}}$ *in the right hand side is strictly larger than the one in the left hand side, which forces the refinement* $pi_1(s) = 2$.

Similarly, in the filtered version of rule 5.24

$$\mathtt{u_2}(\mathtt{int_{out}}(\mathtt{0}, y, xs), y) \to \mathtt{int_{out}}(\mathtt{0}, y, \mathtt{c}(\mathtt{0}, xs))$$

the $\mathtt{int_{out}}$ *subterm appearing in the right hand side is strictly larger that the* $\mathtt{int_{out}}$ *subterm in the left hand side, forcing the refinement* $pi_1(int_{out}) = [1, 2]$, *which precludes a successful termination proof. The filtering of pair 5.40 yields now*

$$\mathtt{U_3}(\mathtt{int_{out}}(x, y), x, y) \to \mathtt{INTLIST_{in}}(zs)$$

where zs *has become an extra variable, forcing the filtering of* $\mathtt{INTLIST_{in}}/1$. *But now the filtering of the dependency pair 5.38 yields*

$$\mathtt{INTLIST_{in}} \to \mathtt{INTLIST_{in}}$$

which cannot be oriented strictly by any reduction pair.

In the approach of [Nishida and Vidal, 2010; Vidal, 2008], one can apply the initial version of π_1 *(since it actually enforces the variable condition on all the pairs* *and rules) to obtain a vanilla DP problem* $\langle \pi_1(\{5.38\}), \pi_1(\mathcal{R}), \mathbf{m} \rangle$ *which can be easily solved by any reduction pair, since thanks to minimality the set of usable rules is empty. Although since the example is neither left-linear nor free of extra variables, this proves nothing in practice. In the approach of [Schneider-Kamp et al., 2009b] one can proceed in the same way, proving termination of infinitary constructor rewriting.*

Before concluding, let us remark that the results must be regarded as orientative only, as we have restricted here to a limited subset of DP processors. Henceforth, these results only represent the *relative* power of these techniques, with regard to this subset.

6

The NARRADAR termination tool

All the automated techniques for proving the termination of narrowing presented in the two previous chapters have been implemented during the course of this thesis. The motivation for producing this implementation was twofold:

- validating of the research performed and the techniques obtained, and
- producing a tool of general applicability which can be reused by other researchers working with narrowing.

Our efforts have crystallised in the termination tool NARRADAR. NARRADAR is a tool based on the dependency framework described in Section 4.1. It implements a number of dependency framework processors, including all the processors introduced in the previous chapters, which enable it to produce proofs of termination for:

- Narrowing,
- Rewriting,
- Relative Rewriting,
- Narrowing from an initial goal,
- Rewriting from an initial goal,
- Relative Rewriting from an initial goal, and
- Logic programs (from an initial goal).

While NARRADAR can tackle termination of rewriting, relative rewriting, and logic programs, it was mainly designed to solve problems of termination of narrowing. Therefore, it is far less powerful than other termination tools intended for proving termination of rewriting and/or logic programs.

6.1 Input

NARRADAR offers a web interface accepting TRSs in the standard TPDB format of the termination problem database[1], with a few extensions to denote narrowing and initial goal problems,

- To demand termination of narrowing, append a section `(STRATEGY NARROWING)`.
- To demand termination from an initial goal, append a section `(STRATEGY GOAL <goal>)`, where goal is either an initial or an abstract goal, depending on whether the problem is rewriting or narrowing.
- It is possible to manually specify the pairs of a DP problem, instead of using the pairs derived from the rules of the problem. For this, append a section `(PAIRS ...)` containing the list of rules denoting the pairs[2] of the problem.
- To enforce minimality when manually entering the pairs of a problem, append a section `(MINIMALITY)`.

These sections can be combined to specify, e.g., an initial goal termination of narrowing problem with a manually defined set of pairs. Note that the ordering of the sections does not matter.

Example 6.1. *Recall the TRS $\mathcal{R}_{Policy}$ from Example 4.1. The following input asks* NARRADAR *to prove termination of narrowing in $\mathcal{R}_{Policy}$:*

```
(VAR src dst x y port s)
(RULES
filter(pckt(src,dst,established)) -> accept
filter(pckt(eth0,dst,new)) -> accept
filter(pckt(address(ip(194,179,1,x),port),dst,new)) ->
  filter(pckt(secure,dst,new))
filter(pckt(address(ip(158,42,x,y),port),dst,new)) ->
  filter(pckt(secure,dst,new))
filter(pckt(secure,address(dst,80),new)) -> accept
filter(pckt(secure,address(dst,other),new)) -> drop
filter(pckt(ppp0,dst,new)) -> drop
filter(pckt(address(ip(123,123,1,1),port),dst,new)) -> accept

pckt(address(ip(10,1,1,1),port),ppp0,s) ->
  pckt(address(ip(123,23,1,1),port),ppp0,s)
pckt(address(ip(10,1,1,2),port),ppp0,s) ->
  pckt(address(ip(123,23,1,1),port),ppp0,s)
```

[1] http://www.lri.fr/ marche/tpdb/format.html

[2] Note that the signature used in the `(RULES..)` section and the signature used in the `(PAIRS..)` section are disjoint and currently there is no way to relate a regular symbol with a tuple symbol. This can be important when introducing initial goal problems with explicit pairs.

```
pckt(src,address(ip(123,123,1,1),port),new) ->
  natroute(pckt(src,address(ip( 10,1,1,1),port),established)
          ,pckt(src,address(ip(10,1,1,2),port),established))

natroute (x,y) -> x
natroute (x,y) -> y
)
(STRATEGY NARROWING)
```

Example 6.2. *Consider the initial goal narrowing problem presented in Example 5.4. We can ask* NARRADAR *to solve it with the input:*

```
(VAR X Y)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X))

(STRATEGY GOAL p_in(g))
(STRATEGY NARROWING)
```

Narradar also accepts logic programs. The syntax for logic programs is a (fairly sensible) subset of prolog syntax, with a one-line comment specifying the initial query in the form:

```
%query: ackermann(i,i,o).
```

where `i` specifies an input (definitely ground) argument and `o` specifies an output (possibly variable) argument.

Example 6.3. *Recall the logic program $\mathcal{P}_{inc}$, introduced in Section 5. We can ask* NARRADAR *to prove termination of this logic program for the initial goal* `p(X)` *with the input:*

```
 p(X) :- q(f(Y)), p(Y).
 q(g(Y)).
 %query: p(o)
```

6.2 Output

Currently NARRADAR only offers two modes of output: textual and schematic representation. Both are available to the user of the web interface.

Example 6.4. *When called with the input shown in Example 6.2,* NARRADAR *takes 40 ms to find a succesful proof. The schematic representation of the proof generated by* NARRADAR *is shown in Figure 6.1, and the textual representation generated is:*

```
Narrowing Problem
TRS: p_in(g(X)) -> u_3(p_in(X),X)
     p_in(X) -> u_1(q_in(f(gen)),X)
     q_in(g(Y)) -> q_out(g(Y))
     u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
     u_2(p_out(Y),X,Y) -> p_out(X)
     u_3(p_out(X),X) -> p_out(g(X))
DPS: p_in#(g(X)) -> p_in#(X)
     p_in#(g(X)) -> u_3#(p_in(X),X)
     p_in#(X) -> q_in#(f(gen))
     p_in#(X) -> u_1#(q_in(f(gen)),X)
     u_1#(q_out(f(Y)),X) -> p_in#(Y)
     u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
GOALS: [abstract p_in#(b)]
PROCESSOR: Finiteness of the following relative termination
           problem implies the termination of narrowing (LOPSTR'09)
  NarrowingGen Rewriting Problem
  TRS: p_in(g(X)) -> u_3(p_in(X),X)
       p_in(X) -> u_1(q_in(f(gen)),X)
       q_in(g(Y)) -> q_out(g(Y))
       u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
       u_2(p_out(Y),X,Y) -> p_out(X)
       u_3(p_out(X),X) -> p_out(g(X))
       GOAL(v2) -> p_in(v2)
  DPS: p_in#(g(X)) -> p_in#(X)
       p_in#(g(X)) -> u_3#(p_in(X),X)
       p_in#(X) -> q_in#(f(gen))
       p_in#(X) -> u_1#(q_in(f(gen)),X)
       u_1#(q_out(f(Y)),X) -> p_in#(Y)
       u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
       GOAL#(v2) -> p_in#(v2)
  GOALS: [concrete GOAL#(v2)]
  TRS': GEN ->= f(GEN)
        GEN ->= g(GEN)
        GEN ->= gen
        GEN ->= p_out(GEN)
        GEN ->= q_out(GEN)
  PROCESSOR: The two systems form a  Hierarchical Combination
             and hence the result from LOPSTR09 applies.
             Finiteness of the following DP problem implies
             relative termination.
      NarrowingGen Rewriting (no minimality) Problem
      TRS: p_in(g(X)) -> u_3(p_in(X),X)
           p_in(X) -> u_1(q_in(f(gen)),X)
           q_in(g(Y)) -> q_out(g(Y))
           u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
           u_2(p_out(Y),X,Y) -> p_out(X)
           u_3(p_out(X),X) -> p_out(g(X))
           GEN -> f(GEN)
```

```
          GEN -> g(GEN)
          GEN -> gen
          GEN -> p_out(GEN)
          GEN -> q_out(GEN)
          GOAL(v2) -> p_in(v2)
     DPS: p_in#(g(X)) -> p_in#(X)
          p_in#(g(X)) -> u_3#(p_in(X),X)
          p_in#(X) -> q_in#(f(gen))
          p_in#(X) -> u_1#(q_in(f(gen)),X)
          u_1#(q_out(f(Y)),X) -> p_in#(Y)
          u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
          GOAL#(v2) -> p_in#(v2)
     GOALS: [concrete GOAL#(v2)]
     PROCESSOR: Dependency Graph Processor (SCCs)
         NarrowingGen Rewriting (no minimality) Problem
         TRS: p_in(g(X)) -> u_3(p_in(X),X)
              p_in(X) -> u_1(q_in(f(gen)),X)
              q_in(g(Y)) -> q_out(g(Y))
              u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
              u_2(p_out(Y),X,Y) -> p_out(X)
              u_3(p_out(X),X) -> p_out(g(X))
              GEN -> f(GEN)
              GEN -> g(GEN)
              GEN -> gen
              GEN -> p_out(GEN)
              GEN -> q_out(GEN)
              GOAL(v2) -> p_in(v2)
         DPS: p_in#(g(X)) -> p_in#(X)
         GOALS: [concrete GOAL#(v2)]
         PROCESSOR: RPO reduction pair
                   The following pairs are strictly decreasing:
                       p_in#(g(X)) -> p_in#(X)
                   The argument filtering used was:
                       f: 1, g: [1], gen: [], p_in: [], p_out: [],
                       q_in: [], q_out: [], u_1: [], u_2: [], u_3: [],
                       GEN: [], GOAL: [], p_in#: [1], q_in#: [],
                       u_1#: [], u_2#: [], u_3#: [], GOAL#: 1
                   The usable rules are:
                   Precedence: g > p_in#
                   Status function:
                   status(g)=LEX with permutation  Nothing
                   status(p_in#)=LEX with permutation  Nothing
             NarrowingGen Rewriting (no minimality) Problem
             TRS: p_in(g(X)) -> u_3(p_in(X),X)
                  p_in(X) -> u_1(q_in(f(gen)),X)
                  q_in(g(Y)) -> q_out(g(Y))
                  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
                  u_2(p_out(Y),X,Y) -> p_out(X)
                  u_3(p_out(X),X) -> p_out(g(X))
                  GEN -> f(GEN)
                  GEN -> g(GEN)
                  GEN -> gen
                  GEN -> p_out(GEN)
                  GEN -> q_out(GEN)
                  GOAL(v2) -> p_in(v2)
             DPS:
             GOALS: [concrete GOAL#(v2)]
             PROCESSOR: We need to prove termination for all the cycles.
                       There are no cycles, so the system is terminating
             RESULT: Problem solved succesfully
```

The graphical representation uses colors to classify the different types of nodes in the dependency graph. *Red* is used to denote an initial pair. Unreachable pairs are shown in *grey*. Pairs not involved in an SCC are shown in *black*. And finally, (the nodes of) every SCC in the graph is rendered in a different colour. In figure 6.1 there is a single SCC, which is rendered in yellow.

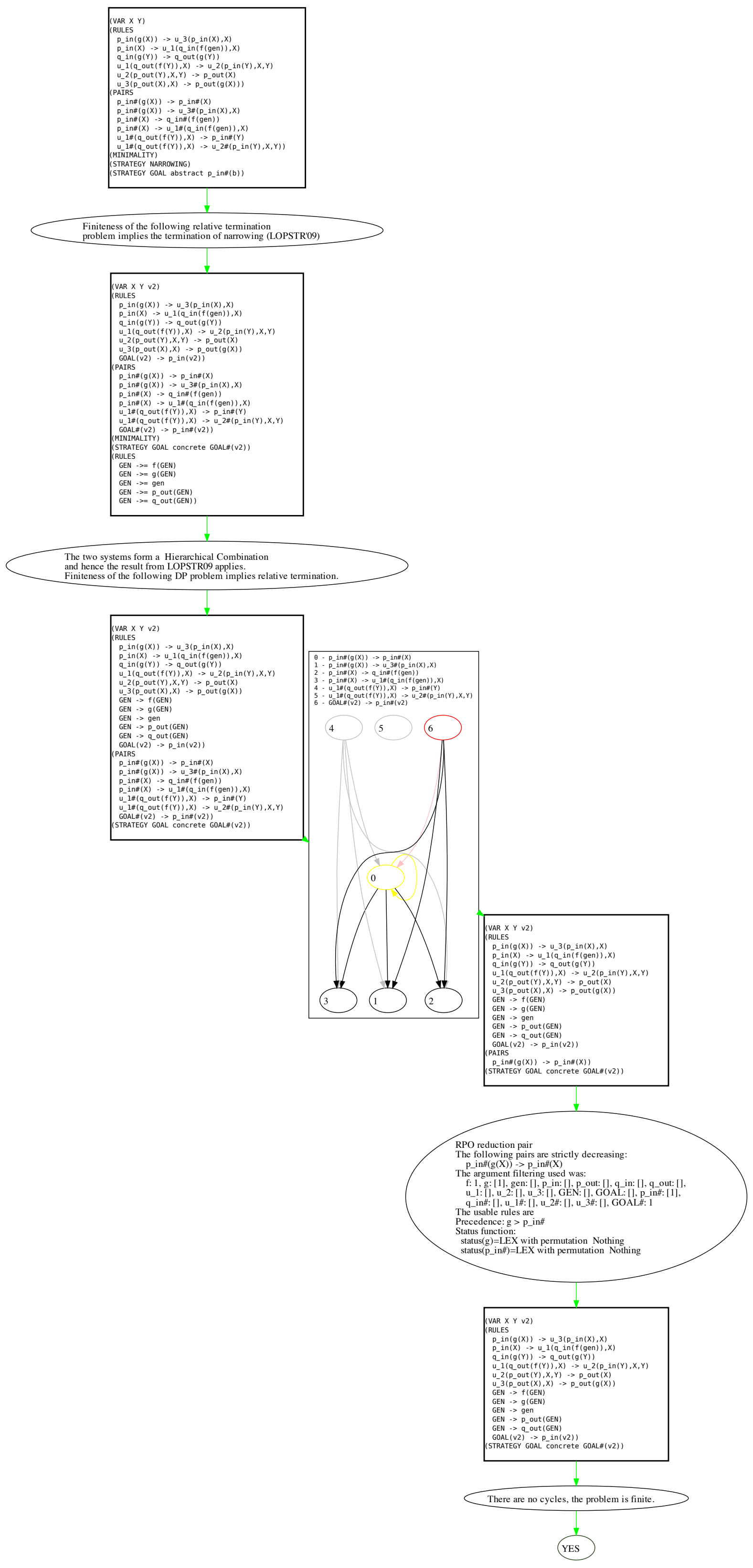
6.3 Implementation details

The implementation consists of around 9.000 lines of Haskell [*Peyton Jones*, 2003], of which

- about 350 are dedicated to the parsing of TPDB input files,
- about 410 are dedicated to the parsing and manipulation of logic programs,
- about 1780 are dedicated to the implementation of term rewriting systems, rewriting and narrowing,
- about 2500 are dedicated to the implementation of the RPO-SAT reduction pair (cf. Section 4.4),
- about 600 are dedicated to producing the output,
- the remaining 3710 implement the DP framework and all the techniques described in Chapters 4 and 5.

The source code is publicly available online[3]. NARRADAR makes use of the modern SMT solver [Yices] to solve the propositional constraints generated by the RPO reduction pair.

[3] `http://github.com/pepeiborra/narradar`



```
(VAR X Y)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X)))
(PAIRS
  p_in#(g(X)) -> p_in#(X)
  p_in#(g(X)) -> u_3#(p_in(X),X)
  p_in#(X) -> q_in#(f(gen))
  p_in#(X) -> u_1#(q_in(f(gen)),X)
  u_1#(q_out(f(Y)),X) -> p_in#(Y)
  u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y))
(MINIMALITY)
(STRATEGY NARROWING)
(STRATEGY GOAL abstract p_in#(b))
```

Finiteness of the following relative termination
problem implies the termination of narrowing (LOPSTR'09)

```
(VAR X Y v2)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X))
  GOAL(v2) -> p_in(v2))
(PAIRS
  p_in#(g(X)) -> p_in#(X)
  p_in#(g(X)) -> u_3#(p_in(X),X)
  p_in#(X) -> q_in#(f(gen))
  p_in#(X) -> u_1#(q_in(f(gen)),X)
  u_1#(q_out(f(Y)),X) -> p_in#(Y)
  u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
  GOAL#(v2) -> p_in#(v2))
(MINIMALITY)
(STRATEGY GOAL concrete GOAL#(v2))
(RULES
  GEN ->= f(GEN)
  GEN ->= g(GEN)
  GEN ->= gen
  GEN ->= p_out(GEN)
  GEN ->= q_out(GEN))
```

The two systems form a Hierarchical Combination
and hence the result from LOPSTR09 applies.
Finiteness of the following DP problem implies relative termination.

```
(VAR X Y v2)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X))
  GEN -> f(GEN)
  GEN -> g(GEN)
  GEN -> gen
  GEN -> p_out(GEN)
  GEN -> q_out(GEN)
  GOAL(v2) -> p_in(v2))
(PAIRS
  p_in#(g(X)) -> p_in#(X)
  p_in#(g(X)) -> u_3#(p_in(X),X)
  p_in#(X) -> q_in#(f(gen))
  p_in#(X) -> u_1#(q_in(f(gen)),X)
  u_1#(q_out(f(Y)),X) -> p_in#(Y)
  u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
  GOAL#(v2) -> p_in#(v2))
(STRATEGY GOAL concrete GOAL#(v2))
```

```
0 - p_in#(g(X)) -> p_in#(X)
1 - p_in#(g(X)) -> u_3#(p_in(X),X)
2 - p_in#(X) -> q_in#(f(gen))
3 - p_in#(X) -> u_1#(q_in(f(gen)),X)
4 - u_1#(q_out(f(Y)),X) -> p_in#(Y)
5 - u_1#(q_out(f(Y)),X) -> u_2#(p_in(Y),X,Y)
6 - GOAL#(v2) -> p_in#(v2)
```

```
(VAR X Y v2)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X))
  GEN -> f(GEN)
  GEN -> g(GEN)
  GEN -> gen
  GEN -> p_out(GEN)
  GEN -> q_out(GEN)
  GOAL(v2) -> p_in(v2))
(PAIRS
  p_in#(g(X)) -> p_in#(X))
(STRATEGY GOAL concrete GOAL#(v2))
```

RPO reduction pair
The following pairs are strictly decreasing:
    p_in#(g(X)) -> p_in#(X)
The argument filtering used was:
    f: 1, g: [1], gen: [], p_in: [], p_out: [], q_in: [], q_out: [],
    u_1: [], u_2: [], u_3: [], GEN: [], GOAL: [], p_in#: [1],
    q_in#: [], u_1#: [], u_2#: [], u_3#: [], GOAL#: 1
The usable rules are
Precedence: g > p_in#
Status function:
  status(g)=LEX with permutation Nothing
  status(p_in#)=LEX with permutation Nothing

```
(VAR X Y v2)
(RULES
  p_in(g(X)) -> u_3(p_in(X),X)
  p_in(X) -> u_1(q_in(f(gen)),X)
  q_in(g(Y)) -> q_out(g(Y))
  u_1(q_out(f(Y)),X) -> u_2(p_in(Y),X,Y)
  u_2(p_out(Y),X,Y) -> p_out(X)
  u_3(p_out(X),X) -> p_out(g(X))
  GEN -> f(GEN)
  GEN -> g(GEN)
  GEN -> gen
  GEN -> p_out(GEN)
  GEN -> q_out(GEN)
  GOAL(v2) -> p_in(v2))
(STRATEGY GOAL concrete GOAL#(v2))
```

There are no cycles, the problem is finite.

YES

Figure 6.1: Proof of Example 6.2

7

Modularity of Basic Narrowing Termination

In Section 3.1 we recalled Hullot's *basic* refinement of narrowing [*Hullot*, 1980] and discussed its termination properties. Basic narrowing is widely used for solving equational unification problems since it is decidable in far more cases than full narrowing while still being complete for equational unification in canonical TRSs.

As already mentioned before, the termination of basic narrowing was established in Hullot's PhD thesis for canonical TRS's, as we saw in Proposition 3.4 at Chapter 3. Namely, basic narrowing terminates in a canonical TRS $\mathcal{R}$ if every derivation stemming from the right-hand side of a rule is finite. In Theorem 3.14 (Section 3.2) we generalized the result above by showing that the canonicity requirement is superfluous.

In this chapter, we ascertain several criteria for modular termination of basic narrowing in hierarchical combinations of TRSs, including generalized proper extensions with a shared subsystem and a novel generalization of this class called *(generalized) relaxed proper extensions*. We assume a standard notion of *modularity*, where a property φ of a TRS is called *modular* if, whenever $\mathcal{R}_1$ and $\mathcal{R}_2$ satisfy φ, then their combination $\mathcal{R}_1 \cup \mathcal{R}_2$ also satisfies φ. Our modularity results for basic narrowing rely on the commutation of basic narrowing sequences, which is studied in Section 3.1.

Structure of the chapter

Section 7.1 studies an interesting connection between basic narrowing and innermost rewriting. Namely, termination of basic narrowing implies termination of innermost rewriting. In Section 7.2 we provide a novel commutation result for basic narrowing derivations. Section 7.3 recalls some standard notions from the modularity of rewriting, and then studies the modularity of basic narrowing termination in the standard classes of modular combinations of TRSs. We demonstrate that termination of basic narrowing is highly modular in some of these classes, including *disjoint unions*, *constructor-sharing unions* and the unions of *composable systems*. The restriction to basic narrowing is of key importance for these results, which do not carry out to unrestricted narrowing. Modularity of basic narrowing termination in hierarchical combinations of TRSs is studied in Section 7.4. Our most general result for the modularity of basic narrowing termination is proved for the novel class of *(generalized) relaxed proper extensions.*

As a direct application, Chapter 8 discusses the modularity of decidability of equational unification via basic narrowing. There, the conditions that guarantee the preservation of termination and completeness of equational unification via basic narrowing in the combination of TRSs are identified, making use of the modularity results presented in this chapter.

The results in this chapter have been published in [*Alpuente et al.*, 2008b].

7.1 Basic narrowing and innermost rewriting

In this section we establish an interesting connection between the termination properties of basic narrowing and those of the rewriting and the innermost rewriting relations.

Basic narrowing [*Hullot*, 1980] is a restriction of narrowing which is essentially based on forbidding narrowing steps on terms brought in by instantiation. In Section 3.1.1 basic narrowing was introduced using the traditional formulation based on the notion of *basic* positions.

In this section however, we will make use of the skeleton-environment formulation, introduced in Section 3.2. In this formulation, the expression to be narrowed is split into a *skeleton* t and an *environment* part θ, i.e., $\langle t, \theta\rangle$. The environment part keeps track of the accumulated substitution so that, at each step, substitutions are composed in the environment part, but are not applied to the expression in the skeleton part, as opposed to ordinary narrowing.

Definition 3.15 (Basic narrowing (skeleton-environment)[*Hölldobler*, 1989])**.** *Given a term $s \in \mathcal{T}(\Sigma, \mathcal{V})$ and a substitution σ, a basic narrowing step for $\langle s, \sigma\rangle$ is defined by $\langle s, \sigma\rangle \overset{b}{\rightsquigarrow}_{p,\mathcal{R},\theta} \langle t, \sigma'\rangle$ if there exist $p \in \mathcal{P}os_\Sigma(s)$, $l \to r \ll \mathcal{R}$, and substitution θ such that $\theta = mgu(s|_p\sigma, l)$, $t = (s[r]_p)$, and $\sigma' = (\sigma\theta)[t]$.*

Along a basic narrowing derivation, the set of *basic* occurrences of each term $t\theta$ occurring in the sequence (given by $\langle t, \theta\rangle$) is $\mathcal{P}os_\Sigma(t)$, and the non-basic occurrences are $\mathcal{P}os_\Sigma(t\theta) - \mathcal{P}os_\Sigma(t)$. When p is not relevant, we simply denote the basic narrowing relation by $\overset{b}{\rightsquigarrow}_{\mathcal{R},\theta}$. By abuse of notation, we also often relax the skeleton-environment notation for basic narrowing steps, i.e., $\langle s, \sigma\rangle \overset{b}{\rightsquigarrow}_{\mathcal{R},\theta} \langle t, \sigma'\rangle$, and use the more compact notation $s\sigma \overset{b}{\rightsquigarrow}_{\mathcal{R},\theta} t\sigma\theta$ instead, but then a suitable track of the basic positions along the narrowing sequences is implicitly done.

We say that $\mathcal{R}$ is $(\overset{b}{\rightsquigarrow})$-terminating (or that basic narrowing terminates in $\mathcal{R}$) when every basic narrowing derivation issuing from any term terminates. All modular termination results in this chapter are based on the termination result for basic narrowing introduced in Theorem 3.14 This termination criterion provides a semi-decidable criteria for $(\overset{b}{\rightsquigarrow})$-termination by just executing the rhs's of the rules. In the literature, the original criterion that all basic narrowing derivations issuing from the rhs's of rules terminate has been approximated by two different sufficient syntactic conditions: every rhs of a rewrite rule is either (i) a constructor term [*Hullot*, 1980] (e.g. a variable) or (ii) a ground term (provided that $\mathcal{R}$ is $(\rightarrow)$-terminating) [*Dershowitz et al.*, 1992]. In the light of the results of Chapter 3, it is obvious that we can consider a third syntactic characterization: every right hand side of a rule is a rigid normal form and $\mathcal{R}$ is confluent.

The following well-known example can be used to show a slightly surprising fact: $(\overset{b}{\rightsquigarrow})$-termination does not entail $(\rightarrow)$-termination.

Example 7.1. *Let us consider the reference example by Toyama [Toyama, 1987] that shows the non-modularity of $(\rightarrow)$-termination for disjoint unions (see [Ohlebusch, 2002]):*

$$\mathcal{R}_0 : \mathtt{f}(0,1,x) \rightarrow \mathtt{f}(x,x,x) \qquad\qquad \mathcal{R}_1 : \mathtt{g}(x,y) \rightarrow x \quad \mathtt{g}(x,y) \rightarrow y$$

Both, $\mathcal{R}_0$ and $\mathcal{R}_1$ are $(\rightarrow)$terminating, whereas the system $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ is not $(\rightarrow)$-terminating as witnessed by

$$\mathtt{f}(0,1,g(0,1)) \rightarrow_{\mathcal{R}} \mathtt{f}(g(0,1),g(0,1),g(0,1)) \rightarrow^*_{\mathcal{R}} \mathtt{f}(0,1,g(0,1)) \rightarrow_{\mathcal{R}} \ldots$$

Regarding basic narrowing, both $\mathcal{R}_0$ and $\mathcal{R}_1$ are $(\overset{b}{\rightsquigarrow})$-terminating since every rhs is unnarrowable, and trivially $\mathcal{R}$ is also $(\overset{b}{\rightsquigarrow})$-terminating, since the right hand sides are still unnarrowable in the disjoint union $\mathcal{R}_0 \cup \mathcal{R}_1$.

The basic narrowing relation and the innermost rewriting relation termination properties are related, as we show in the following.

Definition 7.2 (Innermost restriction)**.** *A reduction step $t \Rightarrow^p t'$ at position p is called* innermost *if no strict subterm of $t|_p$ is reducible by $\Rightarrow$.*

Given a TRS $\mathcal{R}$, we let $\overset{i}{\rightarrow}_{\mathcal{R}}$ denote the innermost restriction of the rewriting relation, or simply innermost rewriting.

Note that the system $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ is innermost terminating[1], i.e. the relation $\overset{i}{\rightarrow}_{\mathcal{R}_0 \cup \mathcal{R}_1}$ is terminating. Actually, by formalizing the idea that every innermost rewriting sequence is a basic sequence [*Hullot*, 1981; *Middeldorp and Hamoen*, 1994; *Yamamoto*, 1988], we are able to prove that $(\overset{b}{\rightsquigarrow})$-termination entails $(\overset{i}{\rightarrow})$-termination, even if does not entail $(\rightarrow)$-termination. To the best of our knowledge, this is the first time that this result is explicit (and exploited) in the technical literature.

In order to formalize this proof, we find it convenient to recall the original definition of basic narrowing based on tracking sets of positions that are enabled for narrowing, as it was introduced in Section 3.1.1.

We introduce an auxiliary notion of basic rewriting derivations, based on the basic positions of Definition 3.2.

[1]Its innermost dependency graph contains no cycles and the results in [*Arts and Giesl*, 2000] apply directly.

Definition 7.3 (Basic Rewriting Derivations). *Given a rewrite derivation*

$$t_0 \xrightarrow{p_0}_{l_0 \to r_0} t_1 \xrightarrow{p_1}_{l_1 \to r_1} \cdots \xrightarrow{p_n}_{l_n \to r_n} t_{n+1}$$

we say it is based on a set of positions B *(or simply called* basic*), with* $B_0, B_1, \ldots, B_n$ *defined as*

$$\begin{aligned} B_0 &= \mathcal{P}os_\Sigma(t_0) \\ B_i &= (B_{i-1} \setminus \{p_i.p \mid p \in \mathcal{P}os(t_{i-1}|_{p_i})\}) \cup p_i.\mathcal{P}os_\Sigma(r_i) \end{aligned}$$

if $B = B_0$*, and* $p_i \in B_i$ *for* $1 \leq i \leq n$*.*

In absence of extra variables, every basic rewriting derivation is also basic narrowing derivation. The following result stating that innermost rewriting derivations are basic was already proved by [*Hullot*, 1981].

Proposition 7.4. [*Hullot*, 1981] *Let* $\mathcal{R}$ *be a TRS,* $t \in \mathcal{T}(\Sigma, \mathcal{V})$ *be a term and* σ *be a normalized substitution. Any innermost rewriting derivation starting from a term* $t\sigma$ *is based on* $\mathcal{P}os_\Sigma(t)$*.*

Hence every innermost rewriting derivation is also a basic narrowing derivation. Now it is immediate to derive the result that termination of basic narrowing entails termination of innermost rewriting, which will be very useful later in Chapter 8.

Proposition 7.5. *Termination of basic narrowing implies termination of innermost rewriting.*

Proof. By contradiction. Let $\mathcal{R}$ be a $(\overset{b}{\leadsto})$-terminating TRS and suppose that there exists an infinite innermost rewriting derivation in $\mathcal{R}$. Since this sequence is basic, it is also a basic narrowing derivation. This contradicts the initial assumption of $\mathcal{R}$ being $(\overset{b}{\leadsto})$-terminating. □

7.2 Commutation of basic narrowing derivations

This section is devoted to study the commutation properties of basic narrowing derivations, which are essential for proving the modularity of basic narrowing termination in Section 7.4.2. The commutation of ordinary narrowing derivations was extensively studied by Rety in [*Réty*, 1987] (cf. Section 3.1.1). We follow the same style here. Let us recall from Chapter 3 (Def. 3.7) the notion of *antecedent* of a position in a rewriting sequence [*Réty*, 1987].

Definition 7.6 (Antecedent of a Position [*Réty*, 1987]). *Let* $t \xrightarrow{p}_{l \to r} t'$ *be a rewriting step,* $v \in \mathcal{P}os(t)$*, and* $v' \in \mathcal{P}os(t')$*. We say position* v *is an* antecedent *of* v' *iff either*

1. $v \parallel p$*, i.e.,* v *and* p *are disjoint, and* $v = v'$*, or*
2. *there exists an occurrence* $u' \in \mathcal{P}os(r)$ *of a variable* x *in* r *s.t.* $v' = p.u'.w$ *and* $v = p.u.w$*, where* $u \in \mathcal{P}os(l)$ *is an occurrence of* x *in* l*.*

With the notations of the previous definition, we have:

1. $t|_v = t'|_{v'}$,
2. v' may have no antecedent if $v' = p.u'$ with $u' \in \mathcal{P}os_\Sigma(r)$, or if $v' < p$,

This notion extends to a rewrite sequence by transitive closure of the rewriting relation in the usual way. As seen in Chapter 3 (Def. 3.9), the notion of antecedent can be extended to narrowing sequences. Below we define it for basic narrowing sequences.

Definition 7.7 (Basic Narrowing Antecedent of a Position [Réty, 1987]). *Let $t \overset{b}{\rightsquigarrow}^*_{\mathcal{R},\sigma} t'$, $v \in \mathcal{P}os(t)$, and $v' \in \mathcal{P}os(t')$. We say v is a* basic narrowing antecedent *of v' iff v is an antecedent of v' in the rewrite sequence $t\sigma \to^*_{\mathcal{R}} t'$.*

Note that if v is an antecedent of v' in the basic narrowing sequence above, then $t|_v\sigma = t'|_{v'}$.

In the following, we consider basic narrowing derivations of the form

$$s \overset{b}{\rightsquigarrow}_{p,g\to d,\sigma} t \overset{b}{\rightsquigarrow}_{q,l\to r,\theta} u \overset{b}{\rightsquigarrow} \dots \tag{7.1}$$

and we are interested in the conditions that allow us to commute the first two steps by first applying to s the rule $l \to r$ and then the rule $g \to d$ to the resulting term. If the subterm $t|_q$ already exists in s, i.e., if q admits at least one antecedent in s, the idea essentially consists in applying $l \to r$ to all the antecedents of q, and then applying $g \to d$ to the resulting term. Let us give an example for motivation, which at the same time illustrates the notion of antecedent.

Example 7.8. [Réty, 1987] *Let us consider the following TRS $\mathcal{R}$:*

$$\mathcal{R} = \left\{ \begin{array}{rcll} \mathtt{f}(x,x) & \to & x & \quad (r_1) \\ \mathtt{g}(x,\mathtt{h}(x)) & \to & x & \quad (r_2) \end{array} \right.$$

and the following basic narrowing derivation:

$$\begin{array}{ll} \langle \mathtt{h}(\mathtt{f}(\mathtt{0},x),\mathtt{g}(x,y)),\{\}\rangle & \overset{b}{\rightsquigarrow}_{p=1,r_1,\{x\mapsto \mathtt{0}\}} \langle \mathtt{h}(x,\mathtt{g}(x,y)),\{x\mapsto \mathtt{0}\}\rangle \\ & \overset{b}{\rightsquigarrow}_{q=2,r_2,\{y\mapsto \mathtt{h}(\mathtt{0})\}} \langle \mathtt{h}(x,x),\{x\mapsto \mathtt{0}\}\rangle. \end{array}$$

The occurrences p and q are disjoint, therefore $q \in \mathcal{P}os\langle \mathtt{h}(x,x),\{x\mapsto \mathtt{0}\}\rangle$ has an antecedent in s at $q_0 = 2$. By first applying r_2 at q_0, and then r_1 at p we get:

$$\begin{array}{ll} \langle \mathtt{h}(\mathtt{f}(\mathtt{0},x),\mathtt{g}(x,y)),\{\}\rangle & \overset{b}{\rightsquigarrow}_{q_0,r_2,\{y\mapsto \mathtt{h}(\mathtt{x})\}} \langle \mathtt{h}(\mathtt{f}(\mathtt{0},x),x),\{\}\rangle \\ & \overset{b}{\rightsquigarrow}_{p,r_1,\{x\mapsto \mathtt{0}\}} \langle \mathtt{h}(x,x),\{x\mapsto \mathtt{0}\}\rangle \end{array}$$

The following result establishes that, in a basic narrowing derivation, the antecedent of a position is always in the skeleton, and case 2 of Definition 7.6 cannot happen.

Lemma 7.9. *Given a basic narrowing derivation* $t \overset{b}{\rightsquigarrow}_{p,l\to r,\sigma} t' \overset{b}{\rightsquigarrow}_{q',g\to d,\theta} u$*, and* $q \in \mathcal{P}os(t)$*, if* q *is an antecedent of* q'*, then* q *and* p *are necessarily disjoint,* q' *is in the skeleton part of* t'*, and* $q = q'$*.*

Proof. Suppose that q and p are not disjoint, then we are in case 2 of Definition 7.6, and $q' = p.u'.w$ for some $u \in Pos_x(r)$. But this means q' is in the environment, and hence the narrowing derivation is not basic, which contradicts the initial assumption. □

In the view of this fact, we can give a simpler definition of the basic narrowing antecedent of a position which is equivalent to the one given in Definition 7.7.

Definition 7.10 (Basic Narrowing Antecedent of a Position)**.** *Let* $t \overset{b}{\rightsquigarrow}_{p,l\to r,} t'$ *be a basic narrowing step, and* v *be a position such that* $v \in \mathcal{P}os(t)$ *and* $v \in \mathcal{P}os(t')$*. We say that* $v \in \mathcal{P}os(t)$ *is a* basic narrowing antecedent *of* $v \in \mathcal{P}os(t)$ *iff* $v \parallel p$*, i.e.,* v *and* p *are disjoint.*

This simpler definition of basic narrowing antecedent leads to the fact that basic narrowing steps can be commuted only when the positions used are disjoint. This is stated in the proposition below. The commutation of basic narrowing sequences is the basis for the modularity results later in Section 7.4.2.

Proposition 7.11 (Commutation of Basic Narrowing)**.** *Every basic narrowing derivation of the form*

$$t \overset{b}{\rightsquigarrow}_{p,g\to d,\sigma_1} (t[d]_p)\sigma_1 \overset{b}{\rightsquigarrow}_{q,l\to r,\sigma_2} u \tag{7.2}$$

where q *admits an antecedent in* t*, i.e.,* p *and* q *are disjoint positions, can be commuted to an equivalent basic narrowing derivation:*

$$t \overset{b}{\rightsquigarrow}_{q,l\to r,\sigma_3} (t[r]_q)\sigma_3 \overset{b}{\rightsquigarrow}_{p,g\to d,\sigma_4} u \tag{7.3}$$

and $\sigma_1\sigma_2 = \sigma_3\sigma_4$*.*

Proof. The following proof is an adaptation of the proof found in [Middeldorp et al., 1996, Lemma 26].

It is useful to make explicit here the skeleton-environment formulation of the derivations involved:

$$\begin{array}{ll} \langle s,\theta\rangle \overset{b}{\rightsquigarrow}_{p,g\to d,\sigma_1} \langle s[d]_p, \theta\sigma_1\rangle \overset{b}{\rightsquigarrow}_{q,l\to r,\sigma_2} \langle s[d]_p[r]_q, \theta\sigma_1\sigma_2\rangle & (7.2) \\ \langle s,\theta\rangle \overset{b}{\rightsquigarrow}_{q,l\to r,\sigma_3} \langle s[r]_q, \theta\sigma_3\rangle \overset{b}{\rightsquigarrow}_{p,g\to d,\sigma_4} \langle s[r]_q[d]_p, \theta\sigma_3\sigma_4\rangle & (7.3) \end{array}$$

where $t = s\theta$ and $u = (s[d]_p[r]_q)\theta\sigma_1\sigma_2 = (s[d]_p[r]_q)\theta\sigma_3\sigma 4$. Note that positions q and p come from the skeleton.

We need to prove that σ_3 and σ_4 exist and that $\sigma_1\sigma_2 = \sigma_3\sigma_4$. Existence of σ_3 follows from the fact that $t|_q$ and l are unifiable.

$$t|_q\sigma_1\sigma_2 = t\sigma_1|_q\ \sigma_2 = (t[d]_p)\sigma_1|_q\ \sigma_2 = l\sigma_2 = l\sigma_1\sigma_2$$

Note that the second step in (7.2) uses the fact that p and q are disjoint positions; the first step formulation in (7.3) follows from (7.2); and the last step uses the fact that l is the lhs of a fresh variant of a rule, and hence $\mathit{Var}(l) \cap D(\sigma_1) = \emptyset$.

So $t|_q$ and l are unifiable; let σ_3 be an idempotent most general unifier of these two terms. By definition, $\sigma_3 \leq \sigma_1\sigma_2$ and there exists ρ such that $\sigma_3\rho = \sigma_1\sigma_2$. We show now that the terms $t[r]_q\sigma_3|_p$ and g are unifiable.

$$(t[r]_q)\sigma_3|_p\ \rho = (t[r]_q|_p)\ \sigma_3\rho = (t[r]_q|_p)\ \sigma_1\sigma_2 = g\sigma_1\sigma_2 = g\sigma_3\rho = g\rho$$

In (7.3), the second step formulation takes advantage of the fact that $\mathit{Var}(g) \cap D(\sigma_3) = \emptyset$. To see why, consider that g is a fresh variant and $D(\sigma_3) \subseteq \mathit{Var}(t|_q) \cup \mathit{Var}(l)$.

By analogous reasoning, since $(t[r]_q)\sigma_3|_p$ and g are unifiable let δ be an idempotent most general unifier, $\delta \leq \rho$. It follows that $\sigma_3\delta \leq \sigma_3\rho$ and hence $\sigma_3\delta \leq \sigma_1\sigma_2$. It should also be clear that $\sigma_3\delta$ is a unifier of $t|_p$ and g, using the fact that $\mathit{Var}(g) \cap D(\sigma_3) = \emptyset$ since $D(\sigma_3) \subseteq \mathit{Var}(t|_p) \cup \mathit{Var}(l)$.

$$t|_p\sigma_3\delta = (t[r]_q|_p)\sigma_3\delta = (t[r]_q)\sigma_3|_p\ \delta = g\delta = g\sigma_3\delta$$

Because σ_1 is a most general unifier for these two terms, $\sigma_1 \leq \sigma_3\delta$, and there is a substitution ρ_2 such that $\sigma_1\rho_2 = \sigma_3\delta$. By considering that $\mathit{Var}(l) \cap D(\sigma_1) = \emptyset$, we conclude

$$(t[d]_q)\sigma_1|_p\ \rho_2 = (t[d]_q|_p)\sigma_1\rho_2 = (t[d]_q|_p)\sigma_3\delta = l\sigma_3\delta = l\sigma_1\rho_2 = l\rho_2$$

Now, since σ_2 is an idempotent most general unifier of $(t[d]_q)\sigma_1|_p$ and l, it follows that $\sigma_2 \leq \rho_2$, and therefore $\sigma_1\sigma_2 \leq \sigma_1\rho_2 = \sigma_3\delta$. But we also have $\sigma_3\delta \leq \sigma_1\sigma_2$, therefore there is a variable renaming τ such that $\sigma_1\sigma_2 = \sigma_3\delta\tau$. Now define $\sigma_4 = \delta\tau$; σ_4 is still a most general unifier of $(t[r]_q)\sigma_3|_p$ and g as required, because most general unifiers are closed under variable renaming [Lassez et al., 1988, Corollary 9]. Hence $\sigma_1\sigma_2 = \sigma_3\sigma_4$, which concludes the proof. □

Example 7.12. [Réty, 1987] *Example 7.12 above showed two basic narrowing steps starting from the term* $\mathtt{h}(\mathtt{f}(\mathtt{0}, x), \mathtt{g}(x, y))$ *which could be commuted with each other. As stated by Proposition 7.11, the positions used,* $p = 1$ *and* $q = 2$*, were indeed disjoint. On the other hand, if we consider the derivation starting from the term* $\mathtt{g}(\mathtt{f}(x, y), z)$*:*

$$\langle \mathtt{g}(\mathtt{f}(x, y), z), \{\} \rangle \overset{b}{\leadsto}_{p=1, r_1, \{y \mapsto x\}} \langle \mathtt{g}(x, z), \{\} \rangle \overset{b}{\leadsto}_{q=\epsilon, r_1, \{z \mapsto x\}}$$

then by Proposition 7.11 there is no way to commute these two basic narrowing steps given at the non-disjoint positions $p = 1$ *and* $q = \epsilon$*.*

7.3 Modularity of Basic Narrowing Termination

In this section we recall the standard classes of modular combinations of TRSs, and discuss the standard notion of $\mathcal{C}_\varepsilon$-termination that we lift to basic narrowing. Then we present the first set of results about the modularity of basic narrowing termination,

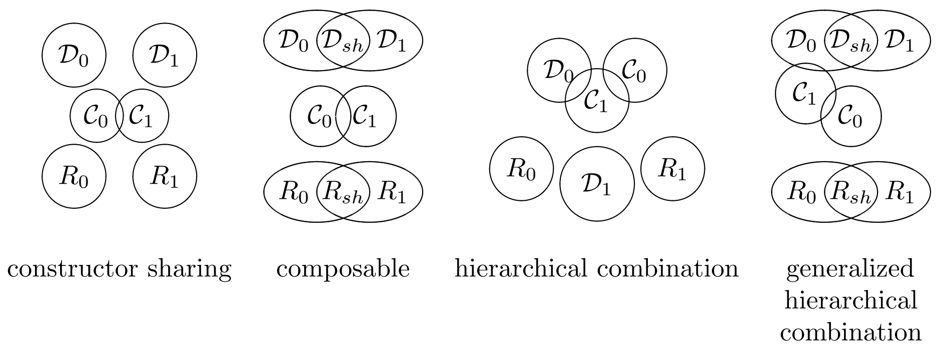


Figure 7.1: Standard modular combinations

including only those combinations in which termination is *always* modular. We discuss the combinations in which modularity requires additional restrictions in Section 7.4.

The following classes of combinations are standard in the literature. Note that we sometimes use the word "union" to denote a combination of two TRSs (sometimes simply called systems). In the case of a hierarchical combination, we sometimes write "an extension system" to denote the combination of two TRSs without clarifying which is the base and which is the extension. We sometimes write $(\mathcal{D} \uplus \mathcal{D}' \uplus \mathcal{C}, R)$ to denote a TRS where $\mathcal{C}$ is the set of constructor symbols and $\mathcal{D} \uplus \mathcal{D}'$ is the set of defined symbols such that $\mathcal{D}$ and $\mathcal{D}'$ are disjoint. We borrow the presentation from [*Ohlebusch*, 2002].

disjoint $\mathcal{R}_0 = (\Sigma_0, R_0)$ and $\mathcal{R}_1 = (\Sigma_1, R_1)$ are *disjoint* if they do not share symbols, i.e., $\Sigma_0 \cap \Sigma_1 = \varnothing$. Their union, called *direct sum*, is denoted $\mathcal{R} = \mathcal{R}_0 \uplus \mathcal{R}_1$.

constructor sharing $(\mathcal{D}_0 \uplus \mathcal{C}_0, R_0)$ and $(\mathcal{D}_1 \uplus \mathcal{C}_1, R_1)$ are *constructor sharing* if they do not share defined symbols, i.e., $\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$.

composable Two systems $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$ are *composable* if $\mathcal{D}_0 \cap \mathcal{D}_1 = \mathcal{D}_0 \cap \mathcal{C}_1 = \mathcal{D}_1 \cap \mathcal{C}_0 = \varnothing$ and both systems share all the rewrite rules that define every shared defined symbol, i.e., $R_{sh} \subseteq R_0 \cap R_1$ where $R_{sh} = \{l \to r \in R_0 \cup R_1 \mid root(l) \in \mathcal{D}_{sh}\}$.

hierarchical combination A system $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ is the *hierarchical combination* (HC) of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{C}_0, R_0)$ and an extension system $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{C}_1, R_1)$ iff $\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$ and $\mathcal{C}_0 \cap \mathcal{D}_1 = \varnothing$.

generalized hierarchical combination A system $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ is the *generalized hierarchical combination* (GHC) of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and an extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$ with a shared subsystem $(\mathcal{F}, R_{sh})$ iff $\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$, $\mathcal{C}_0 \cap \mathcal{D}_1 = \varnothing$, $R_{sh} = R_0 \cap R_1 = \{l \to r \in R_0 \cup R_1 \mid root(l) \in \mathcal{D}_{sh}\}$, and $\mathcal{F} = \{f \in \mathcal{F} \mid f \text{ occurs in } R_{sh}\}$.

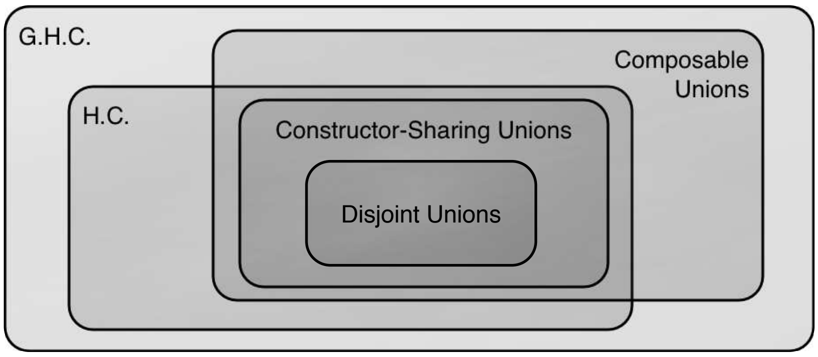


Figure 7.2: Modular combinations

Figure 7.1 shows diagramatic renditions of the definitions introduced above. Roughly speaking, in a hierarchical combination $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ the sets of function symbols defined in $\mathcal{R}_0$ and $\mathcal{R}_1$ are disjoint, and the defined function symbols of the base ($\mathcal{R}_0$) can occur in rules of the extension, but not viceversa. Note that constructor sharing unions generalize disjoint unions, and are themselves generalized by both composable unions and HCs. Finally, these last two combinations are subsumed by GHCs. These relations of subsumption are illustrated in Figure 7.2.

The following example borrowed from [*Ohlebusch*, 2002] illustrates these notions.

Example 7.13. *Consider the following TRSs:*

$$\mathcal{R}_+ = \left\{ \begin{array}{rcl} 0 + y & \to & y \\ \mathtt{s}(x) + y & \to & \mathtt{s}(x+y) \end{array} \right.$$

$$\mathcal{R}_- = \left\{ \begin{array}{rcl} 0 - \mathtt{s}(y) & \to & 0 \\ x - 0 & \to & x \\ \mathtt{s}(x) - \mathtt{s}(y) & \to & x - y \end{array} \right.$$

$$\mathcal{R}_* = \left\{ \begin{array}{rcl} 0 * y & \to & 0 \\ \mathtt{s}(x) * y & \to & (x * y) + y \end{array} \right.$$

$$\mathcal{R}_{pow} = \left\{ \begin{array}{rcl} \mathtt{pow}(x, 0) & \to & \mathtt{s}(0) \\ \mathtt{pow}(x, \mathtt{s}(y)) & \to & x * \mathtt{pow}(x, y) \end{array} \right.$$

$$\mathcal{R}_{app} = \left\{ \begin{array}{rcl} \mathtt{nil} \mathbin{+\!\!+} ys & \to & ys \\ (x : xs) \mathbin{+\!\!+} ys & \to & x : (xs \mathbin{+\!\!+} ys) \end{array} \right.$$

$\mathcal{R}_+$ and $\mathcal{R}_{app}$ are disjoint, $\mathcal{R}_+$ and $\mathcal{R}_-$ are constructor-sharing, $\mathcal{R}_+ \cup \mathcal{R}_*$ is composable with $\mathcal{R}_+ \cup \mathcal{R}_{app}$, and $\mathcal{R}_* \cup \mathcal{R}_+$ is a HC where $\mathcal{R}_*$ extends $\mathcal{R}_+$. Lastly, the system $\mathcal{R}_1 = \mathcal{R}_{pow} \cup \mathcal{R}_+$ extends $\mathcal{R}_0 = \mathcal{R}_* \cup \mathcal{R}_+$ in a GHC with a shared subsystem $\mathcal{R}_{sh} = \mathcal{R}_+$.

As noted by [Rao, 1995a], this classification of combinations of TRSs is straightforwardly applicable to programming languages and incremental program development. The modularity results of direct-sums can be used when two subsystems are defined over different domains, e.g. the natural numbers and the Boolean domain. The modularity results of constructor sharing unions can be used when two subsystems define independent functions (none of the two systems use the procedures defined in the other) over a common domain. HCs model the notion of *modules* in programming languages. In this context, modular termination proofs can pave the way towards applying standard termination analyses and tools to real-life languages since this offers a means to connect termination problems in programming languages to lower-level, rewriting-based solutions. Actually, for proving termination of programs of real-life programming languages supporting a wide range of programming and software development features, termination problems are often step-by-step transformed into "simpler" ones, and then finally translated by rewriting-based termination tools into constraint-solving problems of a given logic which can be handled by standard solvers (see [Durán et al., 2008; Giesl et al., 2006a]).

In the rest of this section, we show that the $(\overset{b}{\rightsquigarrow})$-termination is modular for disjoint unions, constructor-sharing unions, and the union of composable systems.

We start the discussion by extending to basic narrowing the standard notion of $\mathcal{C}_\varepsilon$-termination and proving that $\mathcal{C}_\varepsilon$-termination of basic narrowing is implied by $(\overset{b}{\rightsquigarrow})$-termination.

7.3.1 $\mathcal{C}_\varepsilon$-termination

The notion of $\mathcal{C}_\varepsilon$-termination is used in the literature for proving termination of rewriting in a modular way [Ohlebusch, 2002]. A popular proof technique is essentially based on finding sufficient conditions for the modularity of the more restrictive $\mathcal{C}_\varepsilon$-termination property, which is a sufficient condition for the modularity of $(\rightarrow)$-termination. We recall the notion of $\mathcal{C}_\varepsilon$-termination from page 67 in Section 4.3.

Definition 7.14 ($\mathcal{C}_\mathcal{E}$)**.** [Gramlich, 1994, Def.21] *We let $\mathcal{C}_\mathcal{E}$ denote the TRS*

$$\{\mathtt{c}_\mathcal{E}(x,y) \to x; \mathtt{c}_\mathcal{E}(x,y) \to y\}$$

where $\mathtt{c}_\mathcal{E}$ is a fresh function symbol.

A TRS $\mathcal{R}$ is said to be $\mathcal{C}_\mathcal{E}$-terminating if $\mathcal{R} \cup \mathcal{C}_\mathcal{E}$ is terminating.

Example 7.15 ([Toyama, 1987])**.** *Consider the system $\mathcal{R}$ given by the rule $\mathtt{f}(\mathtt{0},\mathtt{1},x) \to \mathtt{f}(x,x,x)$. $\mathcal{R}$ is terminating, but it is not $\mathcal{C}_\mathcal{E}$-terminating. To see why, consider the following infinite $\to \mathcal{R} \cup \mathcal{C}_\mathcal{E}$ derivation:*

$$\begin{aligned}
\mathtt{f}(\{\underline{\mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})}, \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1}), \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})) &\to_{\mathcal{C}_\mathcal{E}} \\
\mathtt{f}(\{\mathtt{0}, \underline{\mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})}, \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})) &\to_{\mathcal{C}_\mathcal{E}} \\
\underline{\mathtt{f}(\{\mathtt{0}, \mathtt{1}, \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1}))} &\to_{\mathcal{R}} \\
\mathtt{f}(\{\underline{\mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})}, \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1}), \mathtt{c}_\mathcal{E}(\mathtt{0},\mathtt{1})) &\to_{\mathcal{C}_\mathcal{E}} \dots
\end{aligned}$$

This definition can be lifted in the obvious way to (basic) narrowing termination of $\mathcal{R} \uplus \mathcal{C}_\varepsilon$, which we call *$\mathcal{C}_\mathcal{E}$-$(\overset{b}{\rightsquigarrow})$-termination.* The following consequence of Theorem 3.14 is interesting.

Proposition 7.16. *Let $\mathcal{R}$ be a TRS. $\mathcal{R}$ is $\mathcal{C}_\mathcal{E}$-$(\overset{b}{\rightsquigarrow})$-terminating if and only if the TRS $\mathcal{R} \uplus \mathcal{C}_\varepsilon$ is $(\overset{b}{\rightsquigarrow})$-terminating.*

Proof. Immediate by Theorem 3.14 because the rhs's of the rules in $\mathcal{C}_\varepsilon$ satisfy the termination condition for basic narrowing derivations of Theorem 3.14 and $\mathsf{c}_\mathcal{E}$ is a fresh function symbol. □

In contrast, a similar property does not hold for rewriting, but only for innermost rewriting (see [*Ohlebusch*, 2002]). That is, $\mathcal{C}_\varepsilon$-termination is a sufficient condition for $(\rightarrow)$-termination, whereas $\mathcal{C}_\varepsilon$-$(\overset{i}{\rightarrow})$-termination (resp. $\mathcal{C}_\varepsilon$-$(\overset{b}{\rightsquigarrow})$-termination) is a sufficient and necessary condition for $(\overset{i}{\rightarrow})$-termination (resp. $(\overset{b}{\rightsquigarrow})$-termination). This means that both $(\overset{b}{\rightsquigarrow})$-termination and $(\overset{i}{\rightarrow})$-termination are unaffected by non deterministic collapses produced by sets of collapsing rules such as the two rules for g in Example 7.1. As a result, $(\overset{i}{\rightarrow})$-termination is much more modular than $(\rightarrow)$-termination and, as we will see in the following sections, $(\overset{b}{\rightsquigarrow})$-termination is at least as modular as $(\overset{i}{\rightarrow})$-termination.

7.3.2 Disjoint and Constructor-sharing Unions

We start by showing that $(\overset{b}{\rightsquigarrow})$-termination is modular in constructor-sharing unions, a result easily derivable from Theorem 3.14.

Theorem 7.17 (Modularity of Constructor-sharing Unions)**.** *Termination of basic narrowing is a modular property for the union of constructor-sharing systems.*

Proof. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be a constructor-sharing union of two $(\overset{b}{\rightsquigarrow})$-terminating TRSs. We need to prove that any basic narrowing derivation issuing from every right hand side in $\mathcal{R}$ is finite. Every derivation stemming from the right-hand side of a rule of $\mathcal{R}_0$ only uses rules from $\mathcal{R}_0$ (there is no redex w.r.t. $\mathcal{R}_1$), and then such derivations are finite by hypothesis, since $\mathcal{R}_0$ is $(\overset{b}{\rightsquigarrow})$-terminating. The same argument applies to the right-hand sides of the rules of $\mathcal{R}_1$. Finally, by Theorem 3.14, the conclusion follows. □

Obviously, modularity of $(\overset{b}{\rightsquigarrow})$-termination for constructor-sharing unions implies modularity for disjoint unions as well. Example 7.1 illustrates this.

Corollary 7.18 (Modularity of Disjoint Unions)**.** *Termination of basic narrowing is a modular property for the union of disjoint systems.*

7.3.3 Composable unions

Composable unions generalize constructor-sharing unions in that the two systems are allowed to have a set $\mathcal{D}_{sh}$ of shared defined symbols and the rules R_{sh} defining them are all included in both systems. A consequence of this is that the rules in R_{sh} cannot contain calls to functions defined out of R_{sh}.

Lemma 7.19. *Let* $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ *and* $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$ *be two* composable *systems with a shared system* $R_{sh} = \{l \to r \in R_0 \cup R_1 \mid root(l) \in \mathcal{D}_{sh}\}$*. Then no right-hand side of a rule in* R_{sh} *contains a function symbol in* $\mathcal{D}_0 \cup \mathcal{D}_1$*.*

Proof. Let us assume that there is a rule in R_{sh} whose right hand side contains a function symbol $f \in \mathcal{D}_i$ where $i \in \{0, 1\}$. f cannot be a defined symbol in the other system $\mathcal{D}_{1-i}$, since by definition $\mathcal{D}_0 \cap \mathcal{D}_1 = \varnothing$. This implies that f must be a constructor symbol in the other system, i.e., $f \in \mathcal{C}_{1-i}$. But also by definition this is not allowed, as $\mathcal{D}_0 \cap \mathcal{C}_1 = \mathcal{D}_1 \cap \mathcal{C}_0 = \varnothing$. Then f must be a shared symbol, $f \in \mathcal{D}_{sh}$, which contradicts the initial assumption. □

The next theorem extends the modularity of $(\overset{b}{\rightsquigarrow})$-termination to composable systems.

Theorem 7.20 (Modularity of Composable Unions)**.** *Termination of basic narrowing is a modular property of composable systems.*

Proof. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be the union of two composable, $(\overset{b}{\rightsquigarrow})$-terminating TRSs. We can partition the rules in $\mathcal{R}$ in the following disjoint sets:

$$\begin{aligned} \mathcal{R}_{sh} &= \mathcal{R}_0 \cap \mathcal{R}_1 \\ \mathcal{R}'_0 &= \mathcal{R}_0 \setminus \mathcal{R}_{sh} \\ \mathcal{R}'_1 &= \mathcal{R}_1 \setminus \mathcal{R}_{sh} \end{aligned}$$

Basic narrowing derivations issuing from the right-hand sides of the rules in $\mathcal{R}'_0$ or $\mathcal{R}'_1$ terminate by an argument analog to the one used in the proof of Theorem 7.17. By Lemma 7.19, derivations issuing from the right-hand sides of $\mathcal{R}_{sh}$ include no redex w.r.t. either $\mathcal{R}'_0$ or $\mathcal{R}'_1$. Because basic narrowing never reduces any redex occurring at a non-basic position (i.e. in the environment part of the tuple representation), these derivations must also terminate. By Theorem 3.14, the conclusion follows. □

7.4 Modularity in Hierarchical Combinations

So far we have seen that $(\overset{b}{\rightsquigarrow})$-termination is modular without additional conditions for the unions of disjoint systems, constructor-sharing systems and composable systems. For the class of (generalized) hierarchical combinations however, $(\overset{b}{\rightsquigarrow})$-termination is not modular in general, but only under some restrictions. In this section we prove that modularity holds for the novel class of *generalized relaxed proper extensions* (GRPE). *Proper extensions* (PE) are a restriction of hierarchical combinations introduced by

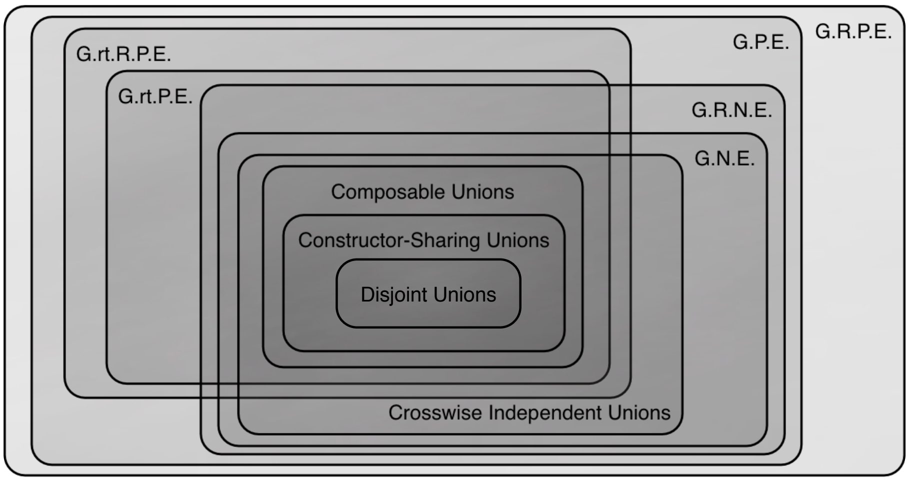


Figure 7.3: More modular combinations

Krishna Rao in [Rao, 1995a] in order to model the modularity of ($\xrightarrow{i}$)-termination, although they have proved useful in other contexts too, see [Ohlebusch, 2002] for an extensive survey. *Generalized proper extensions* (GPE) extend proper extensions to systems with a shared subsystem. Generalized relaxed proper extensions are less restrictive on the right hand sides than generalized proper extensions. For our proof of modularity of basic narrowing termination, we consider generalized relaxed proper extensions and *generalized relaxed nice extensions* (GRNE), two less restrictive versions of generalized proper extensions and *generalized nice extensions* (GNE), respectively. Figure 7.3 shows the different modular combinations studied in this section.

Let us first introduce the auxiliary notion of functional dependency. The function f depends on g, in symbols $f \trianglerighteq g$, if the evaluation of f involves a call to g after one or more rewrite steps.

Definition 7.21 (Dependency relation $\trianglerighteq_{\mathcal{R}}$ [Rao, 1995a]). *For a TRS $(\mathcal{D} \uplus \mathcal{C}, R)$ the dependency relation $\trianglerighteq_{\mathcal{R}}$ is the smallest preorder satisfying the condition $f \trianglerighteq_{\mathcal{R}} g$ whenever there is a rewrite rule $f(s_1, \ldots, s_n) \to r \in R$ and $g(t_1, \ldots, t_n)$ is a subterm of r, with $g \in \mathcal{D}$.*

We often omit $\mathcal{R}$ when it is clear from the context. We say that a symbol $f \in \mathcal{D}$ depends on a symbol $g \in \mathcal{D}$ if $f \trianglerighteq g$.

Definition 7.22 (Split [Rao, 1995a]). *Let $(\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ with an extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. The set $\mathcal{D}_1$ of defined symbols of $\mathcal{R}_1$ is split into two sets $\mathcal{D}_1^0$ and $\mathcal{D}_1^1$ where*

- $\mathcal{D}_1^0 = \{f \in \mathcal{D}_1 \mid \exists g \in \mathcal{D}_0, f \trianglerighteq_{\mathcal{R}} g\}$
- $\mathcal{D}_1^1 = \mathcal{D}_1 \setminus \mathcal{D}_1^0$.

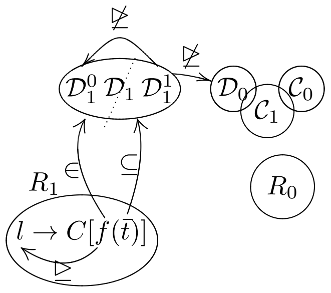


Figure 7.4: Proper extension

That is, $\mathcal{D}_1^0$ contains all the functions from $\mathcal{D}_1$ which depend on functions from $\mathcal{R}_0$, and $\mathcal{D}_1^1$ contains those which do not. We also split R_1 into two subsystems R_1^0 and R_1^1:

- $R_1^0 = \{l \to r \in R_1 \mid root(l) \in \mathcal{D}_1^0\}$
- $R_1^1 = \{l \to r \in R_1 \mid root(l) \in \mathcal{D}_1^1\}$.

We are now ready to introduce the class of generalized proper extensions.

Definition 7.23 (Generalized Proper Extension (GPE) [Rao, 1995a]). *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be a GHC of a base system $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. $\mathcal{R}_1$ is a generalized* proper *extension (GPE) of $\mathcal{R}_0$ iff $R_{sh} \subset R_1^1$ and every rewrite rule $l \to r \in \mathcal{R}_1^0$ satisfies that, for every subterm t of r such that $root(t) \in \mathcal{D}_1^0$ and $root(t) \trianglerighteq_{\mathcal{R}} root(l)$, t contains no function symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

By abusing notation, we often say that $\mathcal{R}_1 \cup \mathcal{R}_0$ is a GPE to mean that it is a GHC where $\mathcal{R}_1$ is a GPE of $\mathcal{R}_0$. This applies not only to GPEs but also to the rest of restrictions of (generalized) hierarchical combinations defined from now on. Figure 7.4 illustrates the notion of *proper extension*, i.e., a generalized proper extension without a common subsystem. The following two examples illustrate the notion of (generalized) proper extensions.

Example 7.24. *Consider computing the factorial of a number in tail recursive style.*

$$\mathcal{R}_! = \left\{ \begin{array}{rcl} \mathtt{fact}(x) & \to & \mathtt{factacc}(x, 1) \\ \mathtt{factacc}(\mathtt{0}, x) & \to & x \\ \mathtt{factacc}(\mathtt{s}(y), x) & \to & \mathtt{factacc}(y, x * \mathtt{s}(y)) \end{array} \right.$$

$$\mathcal{R}_* = \left\{ \begin{array}{rcl} \mathtt{0} * y & \to & \mathtt{0} \\ \mathtt{s}(x) * y & \to & (x * y) + y \end{array} \right.$$

$\mathcal{R}_!$ is a hierarchical extension of $\mathcal{R}_*$, but it is not a GPE, because of the call to the $(*)$ operator below the recursive call to `factacc` *in the right hand side of the 3rd rule of $\mathcal{R}_!$. $(*)$ is an operator defined in the base $\mathcal{R}_*$, and below a recursive call, occurrences of defined symbols of the base system are forbidden by the proper restriction.*

On the other hand, a direct, non tail recursive definition of factorial is a GPE.

Example 7.25. *Consider computing the factorial of a number.*

$$\mathcal{R}_! = \left\{ \begin{array}{rcl} \mathtt{fact}(0) & \to & 1 \\ \mathtt{fact}(s(x)) & \to & \mathtt{s}(x) * \mathtt{fact}(x) \end{array} \right. \quad \mathcal{R}_* = \left\{ \begin{array}{rcl} \mathtt{0} * y & \to & \mathtt{0} \\ \mathtt{s}(x) * y & \to & (x * y) + y \end{array} \right.$$

$\mathcal{R}_!$ is a hierarchical extension of $\mathcal{R}_*$ and it is a GPE, since there are no calls to a defined symbol of the base nested below a recursive call in a right hand side of the extension.

To understand why non-proper extensions can be troublesome for the modularity of $(\overset{b}{\rightsquigarrow})$-termination (and $(\overset{i}{\to})$-termination), consider the following example.

Example 7.26. *Consider the following TRSs, whose combination is hierarchical but not proper since there is a call to* `b` *in the one rule of $\mathcal{R}_1$:*

$$\mathcal{R}_1 : \{\mathtt{f(a)} \to \mathtt{f(b)}\} \qquad\qquad \mathcal{R}_0 : \{\mathtt{b} \to \mathtt{a}\}$$

Individually, basic narrowing is clearly terminating in both systems since their right-hand sides are constants. However, there exists the following infinite $\overset{b}{\rightsquigarrow}_{\mathcal{R}_0 \cup \mathcal{R}_1}$ derivation

$$\mathtt{f(a)} \overset{b}{\rightsquigarrow} \mathtt{f(b)} \overset{b}{\rightsquigarrow} \mathtt{f(a)} \overset{b}{\rightsquigarrow} \cdots$$

produced by the nesting of a redex w.r.t. $\mathcal{R}_0$ inside the recursive call to `f` *in the rhs of the rule of $\mathcal{R}_1$.*

Let us now introduce the main idea behind our *relaxed* generalization of GPEs by means of the following example.

Example 7.27. *Consider the following TRS, an encoding of the exponentiation x^y with group axioms that are commonly used in the specification of many cryptographic protocols [Comon-Lundh, 2004; Cortier et al., 2006], where the constructor symbol* `g` *is used as a generator for the exponentiation.*

$$\mathcal{R}_1 : \quad \mathtt{exp}(\mathtt{exp}(\mathtt{g}, X), Y) \to \mathtt{exp}(\mathtt{g}, X{*}Y)$$
$$\mathcal{R}_0 : \quad X{*}X^{-1} \to \mathtt{1} \qquad X{*}\mathtt{1} \to X \qquad \mathtt{1}{*}X \to X$$

Basic narrowing trivially terminates on each system separately, since every rhs is unnarrowable. However, $\mathcal{R}_1$ is not a GPE of $\mathcal{R}_0$, and even so it is easy to see that basic narrowing does terminate in the union $\mathcal{R}_1 \cup \mathcal{R}_0$. The reason is that the outer function symbol `exp` *in the recursive invocation occurring in the right-hand side of the rule of $\mathcal{R}_1$ is blocked forever, and therefore cannot cause non-termination. Our notion of generalized* relaxed *proper extension captures this idea.*

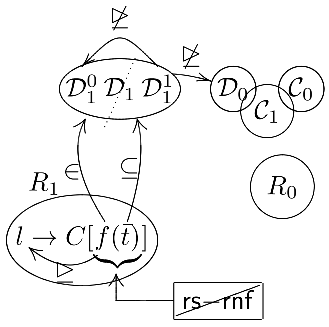


Figure 7.5: Relaxed proper extension

We recall the notion of *root-stable rigid normal form* introduced in Definition 3.29 in Chapter 3, which lifts to narrowing the standard concept of head normal form. By a slight abuse of terminology, in this chapter we consider root-stable rigid normal forms with regard to a basic narrowing relation, instead of the unrestricted narrowing relation. The following definition formalizes this.

Definition 7.28 (Root-Stable Rigid Normal Form for Basic Narrowing)**.** *A term s is a* root-stable rigid normal form *(*rs−rnf*) w.r.t. $\mathcal{R}$ iff either s is a variable or there are no substitutions θ and θ' and terms s' and s'' s.t. $s\theta \overset{>\epsilon *}{\to}_{\mathcal{R}} s' \overset{b}{\leadsto}_{\epsilon,\mathcal{R},\theta'} s''$.*

Roughly speaking, a term t is a root-stable rigid normal form if and only if it is a variable or if there is no possible instantiation of t which enables a basic narrowing step at the root after giving arbitrary rewriting steps below the root. The notion of root-stable rigid normal form is stronger than the notion of rewriting head normal form, i.e., every root-stable rigid normal form is also a head normal form, but the opposite is not true. Constructor terms as well as ground normal forms are trivial cases of root-stable rigid normal forms.

We are ready now to formalize the notion of generalized relaxed proper extensions.

Definition 7.29 (Generalized Relaxed Proper Extension(GRPE))**.** *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. $\mathcal{R}_1$ is a* generalized relaxed proper extension *(GRPE) of $\mathcal{R}_0$ iff $R_{sh} \subset R_1^1$ and every rule $l \to r$ in R_1^0 satisfies the following condition:*

(H1) *for each subterm t of r such that (a) $root(t) \in \mathcal{D}_1^0$, (b) t is not a* rs−rnf*, and (c) $root(t) \trianglerighteq_{\mathcal{R}} root(l)$, t does not contain a function symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

Figure 7.5 illustrates the notion of *relaxed proper extension* rather than generalized relaxed proper extension, i.e., a generalized relaxed proper extension without a common subsystem. The reader can check that the TRS of Example 7.27 is a GRPE.

The most general result of this chapter states that termination of basic narrowing is a modular property for GRPEs.

Corollary 7.30. *Termination of basic narrowing is a modular property for finite generalized relaxed proper extensions.*

The proof of this statement is developed throughout the sections ahead and stated at the end of the chapter. We adapt a succesful proof scheme introduced by Krishna Rao in [*Rao*, 1995a] to study modularity of innermost rewriting termination. This scheme is based on modularly decomposing a proper extension into a number of layered *nice extensions*, a class of hierarchical combinations smaller than proper extensions. Nice extensions are appealing because modularity results that hold in this class can be lifted to proper extensions by means of a suitable modular argument. Informally, modularity of innermost termination is proved for nice extensions as follows:

1. given a base system $\mathcal{R}_0$ and an extension $\mathcal{R}_1$, a set $\mathcal{S}_{\mathcal{R}_0\cup\mathcal{R}_1}$ of terms that contains the right hand sides of the nice extension $\mathcal{R}_0 \cup \mathcal{R}_1$ is identified,

2. innermost termination of a nice extension is proved to be equivalent to innermost termination over the set $\mathcal{S}_{\mathcal{R}_0\cup\mathcal{R}_1}$,

3. a nice extension is split into two TRSs belonging to an even smaller class of modular combinations, called *crosswise independent unions*, and modularity of innermost termination for crosswise independent unions is proved,

4. finally, a commutation result on two specialized relations obtained from $\mathcal{S}_{\mathcal{R}_0\cup\mathcal{R}_1}$ and the modularity of innermost termination for crosswise independent unions are used to prove innermost termination over the set $\mathcal{S}_{\mathcal{R}_0\cup\mathcal{R}_1}$.

In Section 7.4.3, for proving the termination of basic narrowing in generalized relaxed proper extensions, a similar decomposition scheme based on generalized relaxed nice extensions is applied. Then, our proof of modularity of basic narrowing termination for generalized relaxed nice extensions proceeds as follows:

1. given a base system $\mathcal{R}_0$ and an extension $\mathcal{R}_1$, a set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ of terms that contains the right hand sides of the generalized relaxed nice extension $\mathcal{R}_0 \cup \mathcal{R}_1$ is identified (Section 7.4.2),

2. the equivalence between basic narrowing termination for generalized relaxed nice extensions and basic narrowing termination over the set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ follows from Theorem 3.14 (Section 3.1),

3. a generalized relaxed nice extension is split into two crosswise independent systems (Section 7.4.2) and modularity of basic narrowing termination for crosswise independent unions is proved (Section 7.4.1),

4. finally, the commutation result for basic narrowing of Proposition 7.11 (Section 7.2) is applied to two specialized relations obtained from $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ and the modularity of basic narrowing termination for crosswise independent unions is used to prove basic narrowing termination over the set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ (Section 7.4.2).

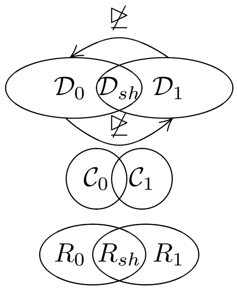


Figure 7.6: Crosswise independent union

7.4.1 Crosswise Independent Unions

Crosswise independent unions (CIU) are a generalization of composable unions in which the non-shared defined symbols of the other system are not allowed in right-hand sides. As for composable unions, defined symbols in $\mathcal{D}_{sh}$ depend only on functions in $\mathcal{D}_{sh}$. We study them as an instrumental step for achieving the proof of modularity of basic narrowing termination for nice extensions in the next section.

Definition 7.31 (Crosswise Independent Union (CIU) [*Rao*, 1995a])**.** [*Rao*, 1995a] *Two TRS's* $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ *and* $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$ *are* crosswise independent *if*

(i) $R_{sh} = R_0 \cap R_1 = \{l \to r \in R_0 \cup R_1 \mid root(l) \in \mathcal{D}_{sh}\}$

(ii) for all $f \in \mathcal{D}_i \cup \mathcal{D}_{sh}$ *and* $g \in \mathcal{D}_{1-i}$, $i \in \{0,1\}$, *we have* $f \not\trianglerighteq_{R_0 \cup R_1} g$.

We say that $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ *is a* crosswise independent union *(CIU) iff* $\mathcal{R}_0$ *and* $\mathcal{R}_1$ *are crosswise independent.*

Figure 7.6 illustrates the notion of a crosswise independent union. Let us show that $(\overset{b}{\leadsto})$-termination is modular for CIUs.

Theorem 7.32. *Termination of basic narrowing is a modular property for crosswise independent unions.*

Proof. Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be the union of two crosswise independent systems. The scheme of the proof is similar to the proof of Theorem 7.20. Essentially, since all the derivations starting from the rhs of a rule in $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ use only rules that belong either to $\mathcal{R}_0$ or $\mathcal{R}_1$, and are, hence, finite in both cases, by Theorem 3.14 every basic narrowing derivation w.r.t. $\mathcal{R}$ is finite. □

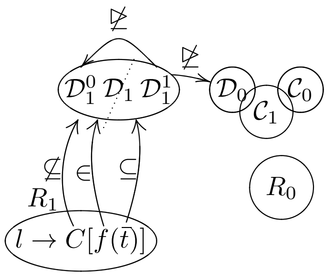


Figure 7.7: Nice extension

7.4.2 Nice Extensions

In this section we prove that $(\overset{b}{\leadsto})$-termination is modular for generalized *relaxed* nice extensions, the relaxed variant of generalized nice extensions introduced by Krishna Rao in [*Rao*, 1995a].

Definition 7.33 (Generalized Nice Extension (GNE) [*Rao*, 1995a])**.** *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. $\mathcal{R}_1$ is a* generalized nice extension *(GNE) of $\mathcal{R}_0$ iff $R_{sh} \subset R_1^1$ and every rule $l \to r \in R_1^0$ satisfies that for each subterm t of r and $root(t) \in \mathcal{D}_1^0$, t contains no function symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

Example 7.25 is a GNE where $\mathcal{R}_!$ is an extension of $\mathcal{R}_*$. Figure 7.7 illustrates a *nice extension* (i.e., a GNE with no shared subsystem).

Note that every GNE is also a GPE by construction. GNEs are a useful restriction of GPEs, because it can be shown that every GPE can be modelled as a *pyramid* of GNEs. The same applies to GRPEs and GRNEs. Therefore in some cases we can lift results from GRNEs to GRPEs.

Definition 7.34 (Generalized Relaxed Nice Extension (GRNE))**.** *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. $\mathcal{R}_1$ is a* generalized relaxed nice extension *(GRNE) of $\mathcal{R}_0$ iff $R_{sh} \subset R_1^1$ and every rule $l \to r \in R_1^0$ satisfies the following condition:*

(N1) *for each subterm t of r such that t is not a* rs−rnf *and $root(t) \in \mathcal{D}_1^0$, t contains no function symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

Example 7.27 shows two systems $\mathcal{R}_1$ and $\mathcal{R}_0$ where $\mathcal{R}_1$ is a GRNE of $\mathcal{R}_0$ but not a GNE nor a GPE.

In a hierarchical combination $\mathcal{R}_0 \cup \mathcal{R}_1$ of a base system $\mathcal{R}_0$ and a GRNE $\mathcal{R}_1$, it is not allowed in the right-hand sides of the extension $\mathcal{R}_1$ to nest calls to functions which depend on the base system $\mathcal{R}_0$, unless the subterm in the right-hand side of $\mathcal{R}_1$ under consideration is a root-stable rigid normal form. This contrasts with the less restricted right-hand sides of proper extensions, where one is allowed to nest function calls that are non rigid and dependent on $\mathcal{R}_0$ as long as they are not mutually recursive.

Following Rao's proof scheme, we prove the result that basic narrowing derivations starting from a special set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ of terms are finite. Let us recall the standard notion of context here. A context is a term C with zero or more 'holes', i.e., the fresh constant symbol $\Box$. If C is a context with k holes and $\bar{t}$ a list of k terms, $C[\bar{t}]$ denotes the result of replacing the k holes in C by the terms in $\bar{t}$.

Definition 7.35 ($\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ terms)**.** *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GRNE of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}^0_1$, $\mathcal{D}^1_1$, R^0_1 and R^1_1 as in Definition 7.22. We define $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ as the set of all terms of the form $C[s_1, \dots, s_n]$, where C is a context in $(\mathcal{D} \cup \mathcal{C})$ and the following conditions hold:*

1. *every subterm s of C with $root(s) \in \mathcal{D}^0_1$ is a $\mathsf{rs-rnf}$ in $\mathcal{R}$.*
2. *for all i, $root(s_i) \in \mathcal{D}^0_1$.*
3. *for all i, s_i is not a $\mathsf{rs-rnf}$ in $\mathcal{R}$.*
4. *s_i contains no function symbol from $\mathcal{D}_0 \cup \mathcal{D}^0_1$ strictly below its root.*

The reader can check that the right-hand sides of the rules in a GRNE belong to the corresponding set of $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$.

By definition, $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ terms have the property that no $\mathcal{R}^0_1$ reduction step is possible within the context C. Also, when we consider the skeleton-environment formulation of basic narrowing, the set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ is skeleton-closed under $\overset{b}{\rightsquigarrow}_{\mathcal{R}_0\cup\mathcal{R}_1}$ if $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ is a GRNE. Lemma 7.36 states this formally.

Lemma 7.36. *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GRNE of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. If $t \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ and either $\langle t, \theta\rangle \overset{b}{\rightsquigarrow}_{\mathcal{R}_1,\sigma} \langle t', \theta\sigma\rangle$ or $\langle t, \theta\rangle \overset{b}{\rightsquigarrow}_{\mathcal{R}_0,\sigma} \langle t', \theta\sigma\rangle$, then $t' \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$.*

Proof. First we consider the closedness of $\mathcal{R}_1$. Let $t = C[s_1, \dots, s_n]$, and let $l \to r \in R_1$ be the rule applied in the basic narrowing step. We consider two cases depending on whether $root(l)$ is in $\mathcal{D}^0_1$ or not.

(a) $root(l) \notin \mathcal{D}^0_1$. That is, $root(l) \in (\mathcal{D}^1_1 \cup \mathcal{D}_{sh})$. There are two subcases:

1. The step took place in C. By definition, no $\mathcal{D}^0_1$ symbol occurs in r, hence t' has the form $C'[t_1, \dots, t_m]$, where each t_i is a subterm of some s_j, $root(t_i) \in \mathcal{D}^0_1$, and no reduction is possible in R^0_1 at a position within the context C'. Since it is always possible to cast t' in this form, the lemma holds.
2. The step took place in a proper subterm of some s_i. By definition $root(r) \in \mathcal{D}^1_1$ and t' has the form $C[s_1, \dots, s'_j, \dots, s_n]$, where $s'_j = s_j[r]_p$ for some position $p > \varepsilon$. Since no D^0_1 symbol occurs in r, s'_j satisfies all the conditions of Definition 7.35 and $t' \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$.

(b) $root(l) \in \mathcal{D}_1^0$. Therefore, the reduction happened at the root of some s_i, since:

- any ocurrence of a $\mathcal{D}_1^0$ symbol in C is a $\mathsf{rs{-}rnf}$ and hence by definition not narrowable.
- s_i itself is not a $\mathsf{rs{-}rnf}$ and hence there exists some instance $s_i\theta$ which is eventually narrowable.
- there are no $\mathcal{D}_1^0$ function symbols below the root of s_i, hence the reduction cannot take place in a proper subterm of s_i.

Let $r = C'[u_1, \ldots, u_m]$; note it is by definition a term in $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$. Then, t' has the form $C[s_1, \ldots, s_{i-1}, r, s_{i+1}, \ldots, s_n]$, and can be written as $C''[s_1, \ldots, s_{i-1}, u_1, \ldots, u_m, s_{i+1}, \ldots, s_n]$, where C'' is the context resulting of *appropriately* joining C and C' so that any subterm r of C'' with $root(r) \in \mathcal{D}_1^0$ is a $\mathsf{rs{-}rnf}$. This means that t' satisfies all the conditions of Definition 7.35, and the lemma holds.

Now let us consider the closedness of $\mathcal{R}_0$. Let $t = C[s_1, \ldots, s_n]$, and $l \to r \in R_0$ be the rule applied in the basic narrowing step. The reduction takes place in C, and since no symbol in $\mathcal{D}_1$ occurs in r, we can write t' as $C'[t_1, \ldots, t_m]$ where:

- C' is the context resulting of replacing a subterm in C with r. By construction, every subterm s in C' with $root(s) \in \mathcal{D}_1^0$ was also in C (since r could not introduce it) and therefore it is by assumption a $\mathsf{rs{-}rnf}$.
- each t_i is a subterm of some s_j with the following properties:
  - $root(t_i) \in \mathcal{D}_1^0$.
  - t_i is not a $\mathsf{rs{-}rnf}$.
- since t_i is a subterm of s_i, the following holds: t_i contains no function symbol in $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.

This concludes the proof. □

Having defined and proved some properties about the set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$, we are now ready to consider $(\overset{b}{\leadsto})$-termination. Concretely, we show that the union of two systems which are $(\overset{b}{\leadsto})$-terminating over $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$ is itself $(\overset{b}{\leadsto})$-terminating. We prove the result by using Proposition 7.11 given in Section 7.2 and some general results on quasi-commutation of abstract relations, which we recall below.

Definition 7.37 (Abstract Reduction System [Bachmair and Dershowitz, 1986])**.** *An abstract reduction system (ARS) is a structure $\mathcal{A} = (A, \{\to_\alpha \mid \alpha \in I\})$ consisting of a set A and a set of binary relations $\to_\alpha$ on A, indexed by a set I. We write $(A, \to_1, \to_2)$ instead of $(A, \{\to_\alpha \mid \alpha \in \{1, 2\}\})$.*

Definition 7.38 (Quasi-commutation [Bachmair and Dershowitz, 1986])**.** *Let $\to_0$ and $\to_1$ be two relations on a set S. The relation $\to_1$ quasi-commutes over $\to_0$ if, for all $s, u, t \in S$ s.t. $s \to_0 u \to_1 t$, there exists $v \in S$ s.t. $s \to_1 v \to_{01}^* t$, where $\to_{01}^*$ is the transitive-reflexive closure of $\to_0 \cup \to_1$.*

Theorem 7.39. [Bachmair and Dershowitz, 1986] *If the relations $\rightarrow_0$ and $\rightarrow_1$ in the ARS(S, $\rightarrow_0$, $\rightarrow_1$) are terminating and $\rightarrow_1$ quasi-commutes over $\rightarrow_0$, then the relation $\rightarrow_0 \cup \rightarrow_1$ is terminating too.*

We now define an ARS with skeleton-environment tuples as elements, where the skeletons come from the set $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$ of terms, and the relationships $\rightarrow_0$ and $\rightarrow_1$ of the ARS are restrictions of basic narrowing.

Definition 7.40 ($\mathcal{A}(\mathcal{R}_0, \mathcal{R}_1)$). *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be a hierarchical combination where $\mathcal{R}_1$ is a GRNE of $\mathcal{R}_0$. Define the sets $\mathcal{D}^0_1$, $\mathcal{D}^1_1$, R^0_1 and R^1_1 as in Definition 7.22. We define the ARS $\mathcal{A}(\mathcal{R}_0, \mathcal{R}_1) = (\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1} \times Subst, \rightarrow_0, \rightarrow_1)$, where the relations $\rightarrow_0$ and $\rightarrow_1$ are defined as follows. Let $s = C[s_0, \ldots, s_n]$ be a term in $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$ below. Then:*

- $$\langle C[s_0, \ldots, s_n], \sigma\rangle \rightarrow_0 \langle C'[u_0, \ldots, u_k], \theta\sigma\rangle$$

  if $\langle C[s_0, \ldots, s_n], \sigma\rangle \overset{b}{\leadsto}_{\mathcal{R}_0 \cup \mathcal{R}^1_1, \theta} \langle C'[u_0, \ldots, u_k], \sigma\theta\rangle$ is a basic narrowing step given within the context C.

- $$\langle C[s_0, \ldots, s_n], \sigma\rangle \rightarrow_1 \langle C[s_0, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n], \theta\sigma\rangle$$

  if $\langle C[s_0, \ldots, s_n], \sigma\rangle \overset{b}{\leadsto}_{\mathcal{R}_1, \theta} \langle C[s_0, \ldots, s_{i-1}, s'_i, s_{i+1}, \ldots, s_n], \theta\sigma\rangle$ is a basic narrowing step given at a subterm s_i, with $i \in [0, \ldots, n]$.

The relation $\rightarrow_1 \cup \rightarrow_0$ is exactly the basic narrowing relation over $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$. In the following we establish that both $\rightarrow_0$ and $\rightarrow_1$ are terminating relations.

Lemma 7.41. *Given the ARS $\mathcal{A}(\mathcal{R}_0,\mathcal{R}_1)$ of Definition 7.40, the relations $\rightarrow_0$ and $\rightarrow_1$ are terminating if $\mathcal{R}_0$ and $\mathcal{R}_1$ are ($\overset{b}{\leadsto}$)-terminating.*

Proof. The relation $\rightarrow_1$ is a subrelation of $\overset{b}{\leadsto}_{\mathcal{R}_1}$, and hence terminating.

On the other hand, $\mathcal{R}_0$ and $\mathcal{R}^1_1$ are crosswise independent: condition (i) in Definition 7.31 is satisfied by construction, and condition (ii) is satisfied since $\mathcal{R}^1_1 \setminus \mathcal{R}_{sh}$ does not depend on $\mathcal{R}_0$ by definition, and neither does $\mathcal{R}_{sh}$ (as $\mathcal{R}_{sh} \subset \mathcal{R}^1_1$ by Definition 7.34). Hence, their union is terminating by Theorem 7.32.

It should be obvious from the definition of $\mathcal{A}$ that $\rightarrow_0$ can be seen as a restriction of the system $\mathcal{R}_0 \cup \mathcal{R}^1_1$ to the terms in $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0 \cup \mathcal{R}_1}$, and therefore termination of $\rightarrow_0$ follows. □

We are now in a position to prove the quasi-commutation of the relation $\rightarrow_1$ over the relation $\rightarrow_0$ in the ARS $\mathcal{A}(\mathcal{R}_0,\mathcal{R}_1)$. The proof of this result relies on Proposition 7.11.

Theorem 7.42. *Given the ARS $\mathcal{A}(\mathcal{R}_0,\mathcal{R}_1)$ of Definition 7.40, the relation $\rightarrow_1$ quasi-commutes over the relation $\rightarrow_0$.*

Proof. We have to show that

$$\forall s,u,t \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1} \text{ s.t. } s \xrightarrow{p}_0 u \xrightarrow{q}_1 t,\ \exists v \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}, p' \in \mathcal{P}os_\Sigma(s) \text{ s.t. } s \xrightarrow{p'}_1 v \to^*_{01} t \tag{7.4}$$

where, by abusing notation, p, q and p' refer to positions that are in the skeleton part of the tuples denoted by s, u, t and v.

As $\to_0 \subseteq \to_{01}$ and $\to_1 \subseteq \to_{01}$, we can equivalently see these as $\to_{01}$ derivations. And now, since $\to_{01}$ is a basic narrowing relation, we can use Proposition 7.11 to prove the result. We only need to show that q admits an antecedent in s. It is easy to see that this is satisfied, because $\to_0$ cannot create new $\to_1$ redexes, which means that redex $u|_q$ was already there in $s|_q$. On the other hand, since no redex can be propagated by basic narrowing, p and q must be disjoint. Applying Proposition 7.11, we have:

$$\forall s,u,t \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1} \text{ s.t. } s \xrightarrow{p}_{01} u \xrightarrow{q}_{01} t,\ \exists v \in \mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1} \text{ s.t. } s \xrightarrow{q}_{01} v \xrightarrow{p}_{01} t \tag{7.5}$$

from which we observe that (1) $p' = q$, (2) t is reached in a single $\to_{01}$ step, and (3) the step $s \to_{01} v$ is indeed given in $\to_1$, which suffices to prove the result. □

We only need to combine Theorem 7.39 and Theorem 7.42 to obtain the desired result on basic narrowing termination of $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ terms.

Corollary 7.43. *Let $\mathcal{R}_1$ and $\mathcal{R}_0$ be two $(\overset{b}{\leadsto})$-terminating systems where $\mathcal{R}_1$ is a GRNE over $\mathcal{R}_0$. Let $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ be a set of terms constructed following definition 7.35. Then every basic narrowing derivation in $\mathcal{R}_0 \cup \mathcal{R}_1$ starting from a term of $\mathcal{S}^{\mathsf{rs-rnf}}_{\mathcal{R}_0\cup\mathcal{R}_1}$ terminates.*

And now by Theorem 3.14, we can derive the modularity of $(\overset{b}{\leadsto})$-termination in generalized relaxed nice extensions.

Corollary 7.44. *Termination of basic narrowing is a modular property for generalized relaxed nice extensions.*

7.4.3 Proof of modularity in proper extensions

In this subsection, we take advantage of the fact that it is possible to model any finite GRPE as a finite pyramid of one or more GRNEs. Essentially, the idea is similar to the modular decomposition of a TRS given in [*Urbain*, 2004]. A given GRPE is reduced to the canonical modular form, a modular partition such that each of the individual modules cannot be split up. In order to achieve this we employ the graph induced by the dependency relation $\trianglerighteq$ on defined function symbols, and the rules corresponding to the symbols of every strongly connected component become a module (i.e., a GRNE).

In order to prove the main result of this section, we first need two auxiliary definitions and one proposition.

Definition 7.45 (Equivalence relation $\approx$ and partial order $\sqsupset$ [Rao, 1995a]). *Let $(\mathcal{D} \uplus \mathcal{C}, R)$ be a generalized hierarchical combination of a base system $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. We define an equivalence relation $\approx_{\mathcal{R}}$ from the dependency relation $\trianglerighteq_{\mathcal{R}}$, where the equivalence class containing f is denoted by $[f]_{\mathcal{R}}$, and a partial ordering $\sqsupset_{\mathcal{R}}$ on the set of equivalence classes:*

- *$f \approx_{\mathcal{R}} g$ iff $f \trianglerighteq_{\mathcal{R}} g$ and $g \trianglerighteq_{\mathcal{R}} f$, where $f, g \in \mathcal{D}_1^0$.*
- *$[f]_{\mathcal{R}} \sqsupset_{\mathcal{R}} [g]_{\mathcal{R}}$ iff $f \trianglerighteq_{\mathcal{R}} g$ and $g \ntrianglerighteq_{\mathcal{R}} f$.*

Since the signature of any TRS is a countable set, the equivalence relation $\approx$ partitions $\mathcal{D}_1^0$ into a countable set E of equivalence classes. Provided that the ordering $\sqsupset$ is noetherian, it can be extended to a well-ordering of type λ, where λ is a countable ordinal.

Definition 7.46 (X_α, S_α). *Let $(\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. For any ordinal α we denote the α-th element in the above well-ordering by E_α (if $\alpha > \lambda$ then $E_\alpha = \varnothing$), and the rules defining its elements as $\mathcal{R}_{E_\alpha} = \{l \to r \in \mathcal{R}_1 \mid root(l) \in E_\alpha\}$. We define the TRS $X_\alpha = \mathcal{R}_{sh} \cup \mathcal{R}_1^1 \cup \mathcal{R}_{E_\alpha}$ and the combined system $S_\alpha = \mathcal{R}_0 \cup (\bigcup_{\beta<\alpha} X_\beta)$. S_0 is $\mathcal{R}_0$ and S_k for $k > \lambda$ is $\mathcal{R}_0 \cup \mathcal{R}_1$.*

The following result establishes the precise relation between GRPEs and GRNEs.

Proposition 7.47. *Let $(\mathcal{D} \uplus \mathcal{C}, R)$ be a GRPE of a base system $(\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $(\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, $\mathcal{R}_1^0$ and $\mathcal{R}_1^1$ as in Definition 7.22. If the relation $\sqsupset_{\mathcal{R}}$ is noetherian, then X_α is a GRNE of S_α for every ordinal α, where S_α and X_α are defined as in Definition 7.46.*

Proof. As $\mathcal{R}_1$ is a GRPE of $\mathcal{R}_0$, we know the shared system R_{sh} is a subset of R_1^1 and hence by construction the first condition in Definition 7.34 is satisfied for any α. We have to show that (N1) holds in X_α. That is, for every $l \to r \in X_\alpha$, if s is a subterm of r s.t. $root(s) \in E_\alpha$ and there is a subterm u of s that contains a defined symbol depending on $Def(S_\alpha) - Def(X_\alpha)$, then s must be a $\mathsf{rs{-}rnf}$. Since $root(s) \in E_\alpha$, it follows that $root(l) \in E_\alpha$ and, by definition, $root(l) \in \mathcal{D}_1^0$ and $root(s) \approx root(l)$. Therefore $root(s) \trianglerighteq root(l)$. Now, since $\mathcal{R}_1$ is a GRPE and $root(s) \in D_1^0$, the following holds by (H1): if s contains a defined symbol depending on $\mathcal{D}_0$, then s is a $\mathsf{rs{-}rnf}$. Finally, since $Def(S_\alpha) - Def(X_\alpha) \subset \mathcal{D}_0 \cup \mathcal{D}_1^0$, X_α is a GRNE of S_α. □

Now, we are ready to establish how GRNEs relate to GRPEs.

Theorem 7.48. *Let $\mathcal{R}_1$ be a finite TRS s.t. it is a GRPE of $\mathcal{R}_0$. $\mathcal{R}_1$ can be seen as a finite pyramid of GRNEs.*

Proof. By Proposition 7.47, assuming that the relation $\sqsupset_{\mathcal{R}_0 \cup \mathcal{R}_1}$ is noetherian. □

With the results developed since Section 7.4.1, the proof of Corollary 7.30 is now a trivial consequence.

Proof of Corollary 7.30. Follows from Corollary 7.44 and Theorem 7.48. □

7.5 Discussion

The completeness and termination properties of basic narrowing have been studied previously in landmark work [*Hullot*, 1980; *Réty*, 1987; *Middeldorp and Hamoen*, 1994]. In this chapter, we contribute to the characterization of basic narrowing termination by providing modularity results for several modular decomposition classes. It turns out that basic narrowing termination is surprisingly modular. In the next Chapter we take advantage of these properties to provide results of modularity of termination *plus* completeness, leading to a study of the modularity of the decidability of equational unification.

A study analog to the one developed in this chapter for basic narrowing would be even more interesting for full unrestricted narrowing, as results of modularity of termination for disjoint unions would provide important theoretical basis for studying the *persistence* of narrowing termination. In the context of many-sorted rewriting, a property is said to be *persistent* if whenever it holds for a many-sorted TRS $\mathcal{R}$, it also holds for the one-sorted version of $\mathcal{R}$ obtained by disregarding type information. In the case of termination, persistence means that termination of a many-sorted TRS $\mathcal{R}$ implies termination of the one-sorted version of $\mathcal{R}$. While persistence of rewriting termination has been widely studied in the literature [*Zantema*, 1994; *Iwami*, 2005a, b; *Aoto*, 1998; *Swiderski et al.*, 2009], as far as I can tell the narrowing case has not been studied yet. Persistence of narrowing termination would have immediate implications for the initial goal termination method of Chapter 5. There, we replaced *logic* variables with *generators*. In a many-sorted setting, we would replace a logic variable of sort S by a generator *of the same sort*. This would lead to a dramatical increase in the precision of the approach. Reusing sort information for proving termination is not an idea of our own; we are in good company [*Zantema*, 1994; *Swiderski et al.*, 2009]. Therefore, we anticipate that studying the modularity properties of full narrowing is a highly relevant line of future work.

8 Modularity of Decidability of Equational Unification

Unification of terms with respect to an equational theory E is the following problem [*Plotkin*, 1970; *Siekmann*, 1989a]:

> Given a set of equations $\Gamma = \{s_1 = t_1, \ldots, s_n = t_n\}_E$, find all substitutions σ (called *solutions*) such that all equations are solved with respect to E, i.e., E implies $s_i\sigma = t_i\sigma$ for all i.

Usually, one is only interested in finding a finite representation of all solutions. One important case of E-unification problem is when the equational theory E can be represented by a canonical set of rewrite rules, where basic narrowing is complete as an equational unification procedure [*Hullot*, 1980]: informally, for every (normalized) solution σ, a more general substitution is computed by narrowing.

Relying on the results of Chapter 7 for the modular termination of basic narrowing, the aim of this chapter is to identify combinations of classes of TRSs where equational unification is decidable.

The results appearing in this chapter have been published in [*Alpuente et al.*, 2010b].

8.1 Existing results

Modularity of unification is a central problem in automated deduction in equational theories that are presented by a finite TRS. In particular, it is known that any disjoint combination of BNT-theories (basic narrowing terminates) is of unification-type finitary [*Schmidt-Schauß*, 1988]. As we will show basic narrowing terminates in the combined theory too, which means that equational unification is decidable in the combined theory (provided this theory additionally satisfies the conditions for the completeness of basic narrowing as a unification procedure).

For the best of our knowledge, modularity of unification in hierarchical combinations of theories has not been previously studied in the related literature. Without loss of generality, we restrict ourselves to the union of two equational theories. The results for the union of several theories can be derived in the obvious way.

As far as we know, [*Prehofer*, 1994] proves the only previous modularity result for the decidability of equational unification (via termination of narrowing) in canonical TRSs. However, this result does not imply the modularity of narrowing termination for a particular class of TRSs but rather the possibility to define a terminating, modular narrowing procedure. The result in [*Prehofer*, 1994] is as follows: *given a canonical TRS $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2$ such that narrowing terminates for $\mathcal{R}_1$ and $\mathcal{R}_2$ and $\mathcal{R}{\downarrow} \subseteq \mathcal{R}_1{\downarrow}\mathcal{R}_2{\downarrow}$ (i.e. the normalization with $\mathcal{R}$ can be obtained by first normalizing with $\mathcal{R}_1$ followed by a normalization with $\mathcal{R}_2$), there is a terminating and complete, modular narrowing strategy for $\mathcal{R}$.* Any complete strategy can be used within the modular procedure given in [*Prehofer*, 1994], including the basic narrowing strategy.

The original completeness result proved by Hullot for canonical TRSs was later generalized in [*Yamamoto*, 1988] by Yamamoto to *weakly innermost normalizing*[1], *weakly canonical*[2] TRSs. Yamamoto also conjectured that weak canonicity would suffice for completeness, but it was refuted by a counterexample in Middeldorp and Hamoen [*Middeldorp and Hamoen*, 1994].

We summarize the existing completeness results for basic narrowing below. But first let us recall the notions of critical pair and orthogonal system. Two (possibly renamed) rules $l \to r$ and $l' \to r'$ *overlap* if there is $p \in \mathcal{P}os_\Sigma(l)$ and substitution σ such that $l|_p\sigma = l'\sigma$. The pair $\langle l\sigma[r'\sigma]_p, r\sigma\rangle$ is called a *critical pair*. A left-linear TRS without critical pairs is called *orthogonal*. Orthogonality implies confluence [*TeReSe*, 2003].

Proposition 8.1 ([*Hullot*, 1980])**.** *Basic narrowing is a complete E-unification algorithm for canonical TRSs.*

Proposition 8.2 ([*Yamamoto*, 1988])**.** *Basic narrowing is a complete E-unification algorithm for weakly canonical, weakly innermost normalizing TRSs.*

Proposition 8.3 ([*Middeldorp and Hamoen*, 1994])**.** *Basic narrowing is a complete E-unification algorithm for weakly normalizing, orthogonal TRSs.*

Proposition 8.4 ([*Middeldorp and Hamoen*, 1994])**.** *Basic narrowing is a complete E-unification algorithm for weakly canonical, right–linear TRSs.*

Note that Proposition 8.2 is strictly more general than Proposition 8.1, since canonicity implies weak canonicity and the property of weak innermost normalization.

Weak normalization (WN) is therefore an important property for completeness. As termination of basic narrowing ($SN^{\overset{b}{\leadsto}}$) implies innermost rewriting termination ($SN^{\overset{i}{\to}}$), which in turn implies weak normalization, we have that termination of basic

[1]A TRS is called *weakly innermost normalizing* if every term has a normal form which can be reached by means of an innermost reduction sequence.

[2]*Weakly canonical* TRSs are confluent & *weakly normalizing* TRSs. A TRS is called *weakly normalizing* if every term has a normal form. Weakly canonicity is referred to as *semicompleteness* in other works, e.g. [*Ohlebusch*, 1995].

narrowing implies weak normalization:

$$SN^{\overset{b}{\rightsquigarrow}} \Rightarrow SN^{\overset{i}{\rightarrow}} \Rightarrow WN$$

Lemma 8.5. *If basic narrowing terminates in $\mathcal{R}$, then $\mathcal{R}$ is weakly normalizing.*

We focus now on termination of the E-unification process via basic narrowing, and its completeness. We say that a narrowing calculus is an *effective E-unification algorithm* for $\mathcal{R}$ if it provides all the solutions in $\mathcal{R}$ for a given E-unification problem or fails when there are no solutions, in a finite amount of time. The following corollary characterizes the conditions under which basic narrowing constitutes an effective E-unification algorithm.

Corollary 8.6. *Basic narrowing is an* effective *E-unification algorithm for $(\overset{b}{\rightsquigarrow})$-terminating confluent TRSs.*

Proof. Termination is one of the assumptions, and completeness follows from Lemma 8.5 and Proposition 8.2. □

It can be seen that this corollary subsumes all the completeness criteria enumerated above as a consequence of the assumption of $(\overset{b}{\rightsquigarrow})$-termination: with respect to Proposition 8.1, termination is dropped; regarding Proposition 8.2, WN and WIN are dropped; since orthogonality implies confluence, Proposition 8.3 is also subsumed; and right–linearity as well as WN from Proposition 8.4 can also be dropped when $(\overset{b}{\rightsquigarrow})$-termination holds.

8.2 Modularity of decidability of E-unification via basic narrowing

In this section, we discuss the modularity of decidability of E-unification via basic narrowing; or more precisely, the preservation of the effectiveness of basic narrowing as a E-unification algorithm in modular combinations of TRSs. Let us start with the case of composable systems.

Theorem 8.7 (Modular Unification in Composable Unions)**.** *Decidability of E-unification via basic narrowing is a modular property for the union of composable $(\overset{b}{\rightsquigarrow})$-terminating, confluent systems.*

Proof. We have shown modularity of basic narrowing termination for composable unions in Theorem 7.20. As for completeness, confluence is shown to be modular for weakly normalizing composable unions in [*Ohlebusch*, 1995, Theorem 5.2], and weak normalization is implied by $(\overset{b}{\rightsquigarrow})$-termination using Lemma 8.5. The result follows from Corollary 8.6. □

Note that the previous result implies also the modularity of E-unification decidability in the unions of disjoint and constructor-sharing ($\overset{b}{\rightsquigarrow}$)-terminating confluent systems.

None of the classes of hierarchical combinations considered previously in this article enjoy modularity of E-unification decidability without additional conditions. The following example shows that modularity of equational unification cannot be directly extended to proper extensions, in this case due to the absence of confluence.

Example 8.8. [*Ohlebusch*, 2002, Example 8.1.3] *Consider the linear, canonical, $\mathcal{C}_\varepsilon$-terminating systems $\mathcal{R}_0$ and $\mathcal{R}_1$.*

$$\mathcal{R}_0 \;:\; \mathtt{a} \;\rightarrow\; \mathtt{b} \qquad \mathcal{R}_1 \;:\; \mathtt{f(a)} \;\rightarrow\; \mathtt{c}$$

$\mathcal{R}_1$ is a proper extension of $\mathcal{R}_0$ but the combined system is not even locally confluent. We have:

$$\mathtt{c} \leftarrow \mathtt{f(a)} \rightarrow \mathtt{f(b)}$$

that is, the term `f(a)` *has two different normal forms.*

Under the additional restriction that the union is an overlay TRS, we prove next that decidability of equational unification is modular for GRPEs. Let us recall the standard notion of *overlay* system. A critical pair $\langle l\sigma[r'\sigma]_p, r\sigma\rangle$ is called an *overlay* if $p = \epsilon$. Given a TRS $\mathcal{R}$, a critical pair $\langle l\sigma[r'\sigma]_p, r\sigma\rangle$ is called *joinable* if there is a term u such that $l\sigma[r'\sigma]_p \rightarrow^*_{\mathcal{R}} u$ and $r\sigma \rightarrow^*_{\mathcal{R}} u$. A TRS whose critical pairs are overlays is called an *overlay TRS*. The proof reuses a number of standard results from the literature, which we include here for self-containment.

Lemma 8.9 (Huet's Critical Pairs Lemma [*Huet*, 1980]). *A term rewriting system is locally confluent if and only f all its critical pairs are joinable.*

Lemma 8.10 (Newman's Lemma [*Newman*, 1942]). *A terminating TRS is confluent if and only if it is locally confluent.*

Lemma 8.11. [*Gramlich*, 1995, Theorem 3.23] *Every locally confluent overlay system is terminating if and only if it is innermost terminating.*

Before formulating and proving the result for GRPEs, let us introduce an auxiliary lemma.

Lemma 8.12. *Local confluence is a modular property of hierarchical combinations of two systems where one is a generalized relaxed proper extension of the other and the union is an overlay system.*

Proof. This result is stated in [*Rao*, 1995a, Lemma 30] for GNEs, and we generalize it here GRPEs.

By the Critical Pairs Lemma, to show local confluence one needs to show joinability of every critical pair. Since every individual system is locally confluent, we only need to consider critical pairs resulting from the overlapping of a symbol from the extension system with a symbol from the base. As the union is an overlay system, overlapping is only possible at the topmost position, and moreover, only in rules from R_{sh} (since $\mathcal{D}_0 \cap \mathcal{D}_1 = \emptyset$). Hence, since these rules are included in both systems, the critical pairs induced are, by assumption, joinable, and by the Critical Pairs Lemma the union system is locally confluent. $\square$

Now we are ready to prove that decidability of E-unification via basic narrowing is modular for GRPEs (and GPEs) if the union is an overlay system.

Theorem 8.13 (Modular Unification in Relaxed Proper Extensions)**.** *Decidability of E-unification via basic narrowing is a modular property for the union of two finite ($\overset{b}{\rightsquigarrow}$)-terminating, confluent systems where one is a generalized relaxed proper extension of the other and the union is an overlay system.*

Proof. Local confluence in the union system follows from Lemma 8.12. Termination of basic narrowing in the union system follows from Corollary 7.30. This implies that the union system is $SN^{\overset{i}{\rightarrow}}$ by Proposition 7.5, and $SN^{\overset{i}{\rightarrow}}$ in a locally confluent overlay system implies termination by Lemma 8.11. Thus, confluence follows from Newman's Lemma. Finally, completeness follows from Corollary 8.6. $\square$

Looking at Example 8.8 now, it can be seen that the union of $\mathcal{R}_0$ and $\mathcal{R}_1$ fails to be an overlay system and therefore the union is not confluent and hence not suitable for E-unification by using basic narrowing.

Though the decidability of equational unification is modular for proper extensions only under the additional condition of overlay system, the following subclass of proper extensions enjoys modularity in *every* case, with no additional restrictions.

Definition 8.14 (Generalized Restricted Proper Extension(GrtPE))**.** [*Rao*, 1995a] *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{R}_1$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, R_1^0 and R_1^1 as in Definition 7.22. $\mathcal{R}_1$ is a* generalized restricted proper extension *(GrtPE) of $\mathcal{R}_0$ if it is a generalized* proper *extension and additionally, no left-hand side of $\mathcal{R}_1$ contains a function symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

Figure 8.1 shows how this new combination is related to previous combinations. Now, we can see that Example 8.8 is not a GrtPE because the extension $\mathcal{R}_1$ contains a defined symbol from the base system $\mathcal{R}_0$ in one left-hand side. On the other hand, in Example 7.25, $\mathcal{R}_!$ is a GrtPE of $\mathcal{R}_*$.

The next theorem states the conditions under which decidability of equational unification is modular for GrtPEs.

Theorem 8.15 (Modular Unification in Restricted Proper Extensions)**.** *Decidability of E-unification via basic narrowing is a modular property for the union of two finite ($\overset{b}{\rightsquigarrow}$)-terminating, confluent systems where one is a generalized restricted proper extension of the other.*

Proof. Basic narrowing termination is modular for finite GrtPEs since it is modular for finite GRPEs by Corollary 7.30. Confluence is shown to be modular for weakly

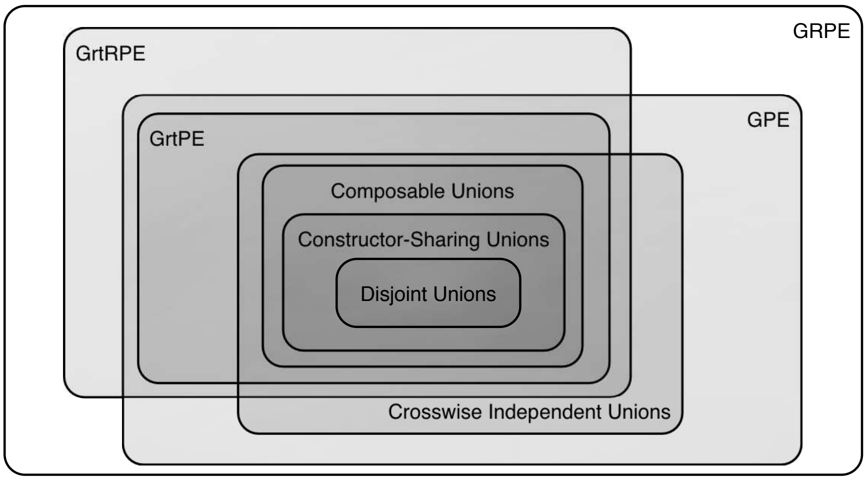


Figure 8.1: Modular combinations for E-unification

normalizing GrtPEs in [*Rao*, 1995b, Theorem 10]. Weak normalization is implied by $(\overset{b}{\rightsquigarrow})$-termination using Lemma 8.5. Finally, completeness follows from Corollary 8.6. □

For the modularity of completeness we also consider a restricted version of relaxed proper extensions called *restricted relaxed proper* extensions. Generalized *restricted* relaxed proper extensions are essentially generalized relaxed proper extensions with the extra condition on the left hand sides introduced in Definition 8.14. In the following we prove that decidability of E-unification is also fully modular for this class of combinations.

Definition 8.16 (Generalized Restricted Relaxed Proper Extension)**.** *Let $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R)$ be a GHC of a base system $\mathcal{R}_0 = (\mathcal{D}_0 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_0, R_0)$ and the extension $\mathcal{R}_1 = (\mathcal{D}_1 \uplus \mathcal{D}_{sh} \uplus \mathcal{C}_1, R_1)$. Define the sets $\mathcal{D}_1^0$, $\mathcal{D}_1^1$, R_1^0 and R_1^1 as in Definition 7.22. $\mathcal{R}_1$ is a* generalized restricted relaxed proper extension *(GrtRPE) of $\mathcal{R}_0$ if and only if it is a generalized relaxed proper extension and additionally, no left-hand side of $\mathcal{R}_1$ contains a funtion symbol from $\mathcal{D}_0 \cup \mathcal{D}_1^0$ strictly below its root.*

Figure 8.1 shows how this new combination is related to previous combinations. The following result states the modularity of E-unification decidability in this class of combinations and strictly subsumes Theorem 8.15.

Theorem 8.17 (Modular Unification in Restricted Relaxed Proper E.)**.** *Decidability of E-unification via basic narrowing is a modular property for the union of two finite $(\overset{b}{\rightsquigarrow})$-terminating, confluent systems where one is a generalized restricted relaxed proper extension of the other.*

TRS	Disjoint	Shared C.	Composable	GrtPE	GrtRPE	GPE	GRPE
Confl	Th. 8.7	Th. 8.7	Th. 8.7	Th. 8.15	Th. 8.17	No(Ex. 8.8)	No (Ex. 8.8)
Confl + Ov	Th. 8.7	Th. 8.7	Th. 8.7	Th. 8.15	Th. 8.17	Th. 8.13	Th. 8.13

Legend

Confl	Confluent
Ov	The combination is an overlay system

Table 8.1: Modularity of Decidability of E-unification via basic narrowing

Proof. Let $\mathcal{R}_1$ and $\mathcal{R}_0$ be two finite $(\overset{b}{\rightsquigarrow})$-terminating confluent TRSs such that $\mathcal{R}_1$ is a GrtRPE of $\mathcal{R}_0$. We define $\mathcal{R}_{1b}$ as the system that results of replacing with a fresh symbol the root symbol in every $\mathtt{rs{-}rnf}$ in a right-hand side of $\mathcal{R}_1$. For instance, for the TRS of Example 7.27 we have:

$$\begin{array}{ll} \mathcal{R}_1 : & \mathtt{exp}(\mathtt{exp}(\mathtt{g}, X), Y) \to \mathtt{exp}(\mathtt{g}, X{*}Y) \\ \mathcal{R}_{1b} : & \mathtt{exp}(\mathtt{exp}(\mathtt{g}, X), Y) \to \mathtt{Q_{exp}}(\mathtt{g}, X{*}Y) \end{array}$$

where $\mathtt{Q_{exp}}$ is a fresh symbol.

$\mathcal{R}_{1b}$ is by construction a GrtPE of $\mathcal{R}_0$, and hence by Theorem 8.15 basic narrowing provides an effective E-unification algorithm for the union system $\mathcal{R}_b = \mathcal{R}_{1b} \cup \mathcal{R}_0$. That is, basic narrowing is complete and terminating in $\mathcal{R}_b$.

In order to show that basic narrowing is complete and terminating in the union $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_0$ too, we proceed by contradiction. Suppose that it is complete and terminating for $\mathcal{R}_b$ but not for $\mathcal{R}$. Then either:

- $\mathcal{R}$ is not $(\overset{b}{\rightsquigarrow})$-terminating,
- $\mathcal{R}$ is not complete.

But neither case is possible, since by definition of $\mathtt{rs{-}rnf}$, $\mathcal{R}_b$ and $\mathcal{R}$ have the same derivations, i.e., one can establish a bisimulation between $\mathcal{R}_b$ and $\mathcal{R}$. Hence basic narrowing is also an effective E-unification algorithm for the union $\mathcal{R}$, which concludes the proof. $\square$

8.3 Discussion

In this section we have presented a number of modularity results for the decidability of E-unification via basic narrowing. Table 8.3 summarizes all these results. Note that $(\overset{b}{\rightsquigarrow})$-termination is an implicit condition in all cases in the table. The hierarchical combination of theories where basic narrowing terminates and is complete as an equational unification procedure was studied. The conditions for the decidability of E-unification under the combinations of theories were identified and characterized. These results are relevant to powerful functional languages featuring narrowing and equational unification, such as Maude (e.g., see [*Clavel et al.*, 2009]), which include effective unification algorithms based on the basic narrowing strategy. Furthermore, in the context of cryptographic protocol verification, equational unification (e.g. via basic narrowing) is the basis for effective security and safety analyses (see [*Escobar et al.*, 2006, 2009]).

9 Conclusion

Narrowing has numerous applications, including equational unification, reachability analyses, termination of programming languages, to name a few. The recent emphasis on equational unification procedures for analysing security protocols [*Escobar et al.*, 2009] is an example of a recent practically oriented research project where narrowing plays a key role for deciding equational unification.

In the mentioned applications and in most of the applications of narrowing, termination of the rules under consideration gives the very strong property of decidability, which is often not only desirable but required. In this dissertation we presented a number of techniques which significantly extend the state of the art in termination of narrowing, and we do so covering a broad spectrum of termination properties, including full termination, termination from an initial goal, and modularity properties.

All these developments not only work in theory but can also be efficiently automated. We presented NARRADAR, a tool that gives the user an automated proof of termination of narrowing where possible. NARRADAR is also quite capable of proving termination of logic programs, with minimal research effort from our part. All these claims are backed by empirical evaluations.

We conclude by pointing out the directions for future work amongst the different topics presented. We started in Chapter 3 with a study of syntactic classes of systems where narrowing always terminates. While we focused on full, unrestricted narrowing, it would be interesting to generalize the study to cover the basic restriction, as its applications in equational unification would surely benefit from a characterization of the systems in which basic narrowing always terminates.

Chapter 4 introduced an automated method for proofs of narrowing termination in term rewriting systems *in general*. Again, the study was focused mainly on full narrowing, and it would be desirable to extend this study to other narrowing variants. In particular, not only basic narrowing, but also innermost and needed narrowing, which are pervasively used in the semantics of functional logic programming languages, should be considered, with the goal of eventually yielding a method for proving the termination of programs. Moreover, we haven't discussed here the problem of *non-termination* of narrowing. Non-termination analysis can help to improve the quality and efficiency of automated termination provers, as pointed out in the related literature [*Payet and Mesnard*, 2006; *Giesl et al.*, 2005a, b; *Geser and Zan-*

tema, 1999; *Thiemann*, 2007]. In the case of narrowing, we anticipate a big efficiency improvement from quickly identifying non-termination problems. However,it is not possible to directly reuse the techniques developed for non-termination of rewriting, since the transformation of a narrowing problem into a rewriting problem developed in Chapter 4 is *sound* but not *complete*. Either a new, complete projection should be developed, or the non-termination analysis be performed directly in the narrowing setting.

Chapter 5 proposed two different extensions to the dependency pair framework, in order to consider initial goal problems as well as relative termination problems. As these two extensions address only the essential aspects required for their application to termination of narrowing from an initial goal, we anticipate considerable room for improvement, including:

- the use of strategies or the Q-restricted rewriting relation of [*Thiemann*, 2007] to more accurately model infinite relative rewriting derivations.
- an extension to restore minimality after the application of the relative termination criterion.

These enhancements would have a direct effect on automated proofs of termination of narrowing from an initial goal. But there is also room for improvement at the narrowing level itself. As already mentioned in the conclusion of Chapter 7, replacing a logic variable with a *universal* generator is an extremely coarse approximation of the values that narrowing could instantiate this logic variable to. Instead, in a many-sorted setting, a logic variable x of *sort* S could be replaced by a specific generator which only produces values of sort S. If the TRS is well-sorted, then this correctly approximates the possible instantiations of x. This technique can be applied to unsorted TRSs too, following the type introduction approach of [*Zantema*, 1994], which employs a simple preprocessing step that, given an unsorted TRS, returns a many-sorted TRS with the most general sort. Overall, this can potentially lead to a much more powerful technique for automated proofs of termination of narrowing from an initial goal. But beforehand, a study of the *persistence* of narrowing termination is required.

As already mentioned in the conclusion of Chapter 7, in the context of many-sorted rewriting a property is said to be *persistent* if whenever it holds for a many-sorted TRS $\mathcal{R}$, it also holds for the unsorted version of $\mathcal{R}$ obtained by disregarding type information. In the case of termination, persistence means that termination of a many-sorted TRS $\mathcal{R}$ implies termination of the unsorted version of $\mathcal{R}$. Persistence of a property P is directly related to the modularity of P for disjoint unions. In Chapter 7, we studied the modularity of termination of basic narrowing for disjoint unions and beyond. Hence these results could lead to results on the persistence of termination of basic narrowing. In the light of this, we identify two possible research directions:

(a) Generalizing the method for termination of narrowing from an initial goal of Chapter 5 to consider also basic narrowing, derive a result on the persistence of termination of basic narrowing from the modularity results of Chapter 7, and combine these two to construct a method for termination of basic narrowing from an initial goal based on type reconstruction.

(b) Proceed by extending the modularity results of Chapter 7 to full narrowing, with the ultimate goal of obtaining a result on the persistence of full narrowing termination, which can then lead to an extension of the technique of Chapter 5 based on type introduction.

Either way, the resulting method has a direct applicability to prove the termination of logic programs. Termination of logic programs is not only of interest for programmers, but has many other applications. For instance, because of its solid theoretical roots, termination of logic programming can lead to proofs of termination for other programming languages, e.g. Java bytecode [*Albert et al.*, 2008]. A look to the literature reveals that termination of logic programs is a highly active research field: [*Schneider-Kamp et al.*, 2009b; *Schneider-Kamp*, 2008; *Schneider-Kamp et al.*, 2009a; *Nguyen et al.*, 2009; *Genaim and Codish*, 2003; *Codish and Genaim*, 2003; *Bruynooghe et al.*, 2007; *Codish and Taboch*, 1999; *Arts and Zantema*, 1996; *Marchiori*, 1996; *Colussi et al.*, 1995]. Even so, our empirical results in Chapter 5 suggest that an approach based on the termination of narrowing could outperform the currently most succesful[1] approach, [*Schneider-Kamp et al.*, 2009b]. This is only mildly surprising, as there is a very direct connection between narrowing and the operational semantics of logic programming [*Bosco et al.*, 1988].

[1]as per the Termination Competition edition of 2009.

Bibliography

Alarcón, B., R. Gutiérrez, J. Iborra, and S. Lucas, Proving termination of context-sensitive rewriting with Mu–Term, *Electr. Notes Theor. Comput. Sci.*, *188*, 105–115, 2007.

Albert, E., P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini, Termination analysis of java bytecode, in *FMOODS*, *Lecture Notes in Computer Science*, vol. 5051, edited by G. Barthe and F. S. de Boer, pp. 2–18, Springer, 2008.

Alpuente, M., M. Falaschi, M. Gabbrielli, and G. Levi, The semantics of equational logic programming as an instance of CLP, in *Logic Programming Languages: Constraints, Functions and Objects*, edited by K. R. Apt, J. W. de Bakker, and J. J. M. M. Rutten, pp. 49–81, The MIT Press, Cambridge, Massachussets, USA, 1993.

Alpuente, M., M. Falaschi, and G. Vidal, Compositional Analysis for Equational Horn Programs, in *4th Int'l Conf. on Algebraic and Logic Programming, ALP'94*, *LNCS*, vol. 850, pp. 77–94, Springer, 1994.

Alpuente, M., M. Falaschi, and G. Levi, Incremental Constraint Satisfaction for Equational Logic Programming, *Theoretical Computer Science*, *142*(1), 27–57, 1995a.

Alpuente, M., M. Falaschi, and F. Manzo, Analyses of Unsatisfiability for Equational Logic Programming, *Journal of Logic Programming*, *22*(3), 221–252, 1995b.

Alpuente, M., S. Escobar, and J. Iborra, Termination of Narrowing Using Dependency Pairs, in *Proc. of the 24th International Conference on Logic Programming (ICLP 2008)*, *Lecture Notes in Computer Science*, vol. 5366, edited by M. G. de la Banda and E. Pontelli, pp. 317–331, Springer-Verlag, 2008a.

Alpuente, M., S. Escobar, and J. Iborra, Modular termination of basic narrowing, in *19th Int'l Conference on Rewriting Techniques and Applications (RTA)*, Lecture Notes in Computer Science, to appear, Springer, 2008b.

Alpuente, M., S. Escobar, and J. Iborra, Termination of narrowing revisited, *Theoretical Computer Science*, *410*(46), 4608 – 4625, doi:DOI:10.1016/j.tcs.2009.07.037, abstract Interpretation and Logic Programming: In honor of professor Giorgio Levi, 2009.

Alpuente, M., M. Comini, S. Escobar, M. Falaschi, and J. Iborra, A compact fixpoint semantics for term rewriting systems, *Theoretical Computer Science*, *In Press*, doi: 10.1016/j.tcs.2010.05.012, 2010a.

Alpuente, M., S. Escobar, and J. Iborra, Modular termination of basic narrowing and equational unification, *Logic Journal of the IGPL*, special issue on unification algorithms, to appear, 2010b.

Antoy, S., and Z. M. Ariola, Narrowing the Narrowing Space, in *Int'l Symposium on Programming Language Implementation and Logic Programming (PLILP)*, *Lecture Notes in Computer Science*, vol. 1292, pp. 1–15, Springer-Verlag, 1997.

Antoy, S., and M. Hanus, Overlapping Rules and Logic Variables in Functional Logic Programs, in *Proc. of the 22snd Int'l Conf. on Logic Programming (ICLP'06)*, pp. 87–101, Springer LNCS 4079, 2006.

Aoto, T., Solution to the problem of zantema on a persistent property of term rewriting systems, in *Proc. 7th International Conf. on Algebraic and Logic Programming, LNCS, 1490*, pp. 250–265, Springer-Verlag, 1998.

Apt, K. R., *From Logic Programming to PROLOG*, 328+viii pp., Prentice-Hall, Englewood Cliffs, NJ, 1997.

Arts, T., and J. Giesl, Termination of Term Rewriting using Dependency Pairs, *Theoretical Computer Science*, *236*(1-2), 133–178, 2000.

Arts, T., and H. Zantema, Termination of Logic Programs Using Semantic Unification, in *Int'l Workshop on Logic-based Program Synthesis and Transformation*, *Lecture Notes in Computer Science*, vol. 1048, pp. 219–233, Springer-Verlag, 1996.

Baader, F., and T. Nipkow, *Term Rewriting and All That*, Cambridge University Press, 1998.

Bachmair, L., and N. Dershowitz, Commutation, transformation, and termination, *8th Int'l Conference on Automated Deduction*, 1986.

Bosco, P., E. Giovanetti, and C. Moiso, Narrowing vs. sld-resolution, *Theor. Comput. Sci.*, *59*(1-2), 3–23, doi:http://dx.doi.org/10.1016/0304-3975(88)90095-3, 1988.

Bruynooghe, M., M. Codish, J. P. Gallagher, S. Genaim, and W. Vanhoof, Termination analysis of logic programs through combination of type-based norms, *ACM Trans. Program. Lang. Syst.*, *29*(2), 2007.

Chabin, J., and P. Réty, Narrowing directed by a graph of terms, in *4th Int'l Conference on Rewriting Techniques and Applications (RTA)*, *Lecture Notes in Computer Science*, vol. 4581, pp. 112–123, Springer-Verlag, 1991.

Christian, J., Some termination criteria for narrowing and e-narrowing, in *11th Int'l Conf. on Automated Deduction CADE'92*, *LNCS*, vol. 607, pp. 582–588, Springer, 1992.

Clavel, M., F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott, *All About Maude - A High-Performance Logical Framework: How to Specify, Program, and Verify Systems in Rewriting Logic*, *Lecture Notes in Computer Science*, vol. 4350, Springer-Verlag, 2007.

Clavel, M., F. Durán, S. Eker, S. Escobar, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. L. Talcott, Unification and narrowing in maude 2.4, in *Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings*, *Lecture Notes in Computer Science*, vol. 5595, edited by R. Treinen, pp. 380–390, Springer, 2009.

Codish, M., and S. Genaim, Proving termination one loop at a time, in *WLPE*, *Report*, vol. CW371, edited by F. Mesnard and A. Serebrenik, pp. 48–59, Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, B-3001 Heverlee (Belgium), 2003.

Codish, M., and C. Taboch, A semantic basis for the termination analysis of logic programs, *J. Log. Program.*, *41*(1), 103–123, 1999.

Codish, M., V. Lagoon, and P. J. Stuckey, Solving partial order constraints for lpo termination, *CoRR*, *abs/cs/0512067*, 2005.

Codish, M., P. Schneider-Kamp, V. Lagoon, R. Thiemann, and J. Giesl, Sat solving for argument filterings, in *LPAR*, *LNCS*, vol. 4246, edited by M. Hermann and A. Voronkov, pp. 30–44, Springer, 2006.

Colussi, L., E. Marchiori, and M. Marchiori, On termination of constraint logic programs, in *CP*, *Lecture Notes in Computer Science*, vol. 976, edited by U. Montanari and F. Rossi, pp. 431–448, Springer, 1995.

Comon, H., M. Haberstrau, and J.-P. Jouannaud, Syntacticness, cycle-syntacticness, and shallow theories, *Information and Computation*, *111*(1), 154–191, 1994.

Comon-Lundh, H., Intruder Theories (Ongoing Work), in *7th Int'l Conference on Foundations of Software Science and Computation Structures (FOSSACS)*, *Lecture Notes in Computer Science*, vol. 2987, pp. 1–4, Springer-Verlag, 2004.

Cortier, V., S. Delaune, and P. Lafourcade, A Survey of Algebraic Properties used in Cryptographic Protocols, *Journal of Computer Security*, *14*(1), 1–43, 2006.

Damas, L., and R. Milner, Principal type-schemes for functional programs, in *POPL '82: Proceedings of the 9th ACM SIGPLAN-SIGACT symposium on Principles of programming languages*, pp. 207–212, ACM, New York, NY, USA, doi:http://doi.acm.org/10.1145/582153.582176, 1982.

de Dios-Castro, J., and F. López-Fraguas, Extra Variables Can Be Eliminated from Functional Logic Programs, in *Proc. of the 6th Spanish Conf. on Programming and Languages (PROLE'06)*, pp. 3–19, ENTCS 188, 2007.

Dershowitz, N., Termination of Rewriting, *Journal of Symbolic Computation*, *3*(1/2), 69–116, 1987.

Dershowitz, N., Goal Solving as Operational Semantics, in *Int'l Logic Programming Symposium, ILPS'95*, pp. 3–17, MIT Press, Cambridge, MA, 1995.

Dershowitz, N., Termination dependencies, in *Proc. of the 6th Int'l Workshop on Termination*, Technical Report DSIC-II/15/03, pp. 27–30, 2003.

Dershowitz, N., and S. Mitra, Jeopardy, in *10th Int'l Conference on Rewriting Techniques and Applications (RTA)*, *Lecture Notes in Computer Science*, vol. 1631, edited by P. Narendran and M. Rusinowitch, pp. 16–29, Springer, 1999.

Dershowitz, N., and G. Sivakumar, Goal-directed Equation Solving, in *7th National Conference on Artificial Intelligence*, pp. 166–170, Morgan Kaufmann, 1988.

Dershowitz, N., S. Mitra, and G. Sivakumar, Decidable Matching for Convergent Systems, in *11th Int'l Conference on Automated Deduction (CADE)*, *Lecture Notes in Computer Science*, vol. 607, edited by D. Kapur, pp. 589–602, Springer-Verlag, Berlin, 1992.

Durán, F., S. Lucas, and J. Meseguer, MTT: The Maude termination tool (system description), in *Automated Reasoning, 4th International Joint Conference, IJCAR 2008, Sydney, Australia, August 12-15, 2008, Proceedings*, *Lecture Notes in Computer Science*, vol. 5195, edited by A. Armando, P. Baumgartner, and G. Dowek, pp. 313–319, Springer, 2008.

Endrullis, J., J. Waldmann, and H. Zantema, Matrix interpretations for proving termination of term rewriting, *J. Autom. Reason.*, *40*(2-3), 195–220, doi:http://dx.doi.org/10.1007/s10817-007-9087-9, 2008.

Escobar, S., and J. Meseguer, Symbolic model checking of infinite-state systems using narrowing, in *18th Int'l Conference on Rewriting Techniques and Applications, RTA 2007*, *Lecture Notes in Computer Science*, vol. 4533, pp. 153–168, Springer-Verlag, 2007.

Escobar, S., C. Meadows, and J. Meseguer, A Rewriting-Based Inference System for the NRL Protocol Analyzer and its Meta-Logical Properties, *Theoretical Computer Science*, *367*(1-2), 162–202, 2006.

Escobar, S., C. Meadows, and J. Meseguer, Maude-NPA: Cryptographic protocol analysis modulo equational properties, in *FOSAD 2008/2009 Tutorial Lectures*, *Lecture Notes in Computer Science*, vol. 5705, edited by A. Aldini, G. Barthe, and R. Gorrieri, pp. 1–50, Springer, 2009.

Fay, M., First-Order Unification in an Equational Theory, in *4th Int'l Conference on Automated Deduction, CADE'79*, pp. 161–167, 1979.

Genaim, S., and M. Codish, Inferring termination conditions for logic programs using backwards analysis, *CoRR*, *cs.PL/0312023*, 2003.

Geser, A., and H. Zantema, Non-looping string rewriting, *ITA*, *33*(3), 279–302, 1999.

Giesl, J., T. Arts, and E. Ohlebusch, Modular termination proofs for rewriting using dependency pairs, *J. Symb. Comput.*, *34*(1), 21–58, 2002.

Giesl, J., R. Thiemann, P. Schneider-Kamp, and S. Falke, Automated termination proofs with AProVe, in *Proc. 15th Int'l Conf. on Rewriting Techniques and Applications, RTA'04*, LNCS, pp. 210–220, 2004.

Giesl, J., R. Thiemann, and P. Schneider-Kamp, Proving and disproving termination in the dependency pair framework, in *Deduction and Applications*, *Dagstuhl Seminar Proceedings*, vol. 05431, edited by F. Baader, P. Baumgartner, R. Nieuwenhuis, and A. Voronkov, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2005a.

Giesl, J., R. Thiemann, and P. Schneider-Kamp, Proving and disproving termination of higher-order functions, in *FroCos*, *Lecture Notes in Computer Science*, vol. 3717, edited by B. Gramlich, pp. 216–231, Springer, 2005b.

Giesl, J., R. Thiemann, and P. Schneider-Kamp, The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs, in *Proc. of the 11th Int'l Conf. on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'04)*, pp. 301–331, Springer LNCS 3452, 2005c.

Giesl, J., S. Swiderski, P. Schneider-Kamp, and R. Thiemann, Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages, in *Proc. of the 17th International Conference on Term Rewriting and Applications (RTA 2006)*, edited by F. Pfenning, pp. 297–312, Springer LNCS 4098, 2006a.

Giesl, J., R. Thiemann, P. Schneider-Kamp, and S. Falke, Mechanizing and Improving Dependency Pairs, *Journal of Automated Reasoning*, *37*(3), 155–203, 2006b.

Gramlich, B., Generalized sufficient conditions for modular termination of rewriting, *Appl. Algebra Eng. Commun. Comput.*, *5*, 131–158, 1994.

Gramlich, B., Abstract relations between restricted termination and confluence properties of rewrite systems, *Fundamenta Informaticae*, *24*, 3–23, 1995.

Hamza, M. H. (Ed.), *IASTED International Conference on Artificial Intelligence and Applications, part of the 23rd Multi-Conference on Applied Informatics, Innsbruck, Austria, February 14-16, 2005*, IASTED/ACTA Press, 2005.

Hanus, M., The Integration of Functions into Logic Programming: From Theory to Practice, *Journal of Logic Programming*, *19&20*, 583–628, 1994.

Hirokawa, N., and A. Middeldorp, Dependency pairs revisited, in *Proc. 15th Int'l Conf. on Rewriting Techniques and Applications, RTA'04*, *LNCS*, vol. 3091, pp. 249–268, Springer, 2004.

Hirokawa, N., and A. Middeldorp, Automating the Dependency Pair Method, *Inf. Comput.*, *199*(1-2), 172–199, 2005.

Hölldobler, S., *Foundations of Equational Logic Programming*, *Lecture Notes in Artificial Intelligence*, vol. 353, Springer-Verlag, Berlin, 1989.

Huet, G. P., Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems, *Journal of the ACM*, *27*(4), 797–821, 1980.

Hullot, J.-M., Canonical Forms and Unification, in *5th Int'l Conference on Automated Deduction CADE'80*, *LNCS*, vol. 87, pp. 318–334, Springer-Verlag, Berlin, 1980.

Hullot, J.-M., Compilation de Formes Canoniques dans les Théories q́uationelles. Thése de Doctorat de Troisième Cycle, Ph.D. thesis, Université de Paris Sud, Orsay (France), 1981.

Iborra, J., N. Nishida, and G. Vidal, Goal directed and relative dependency pairs for the termination of narrowing, in *Proc. of the 19th Int'l Workshop on Logic-Based Program Synthesis and Transformation (LOPSTR 2009), Revised Selected Papers*, *Lecture Notes in Computer Science*, vol. 6037, edited by D. De Schreye, Springer-Verlag, 2010.

Iwami, M., Persistence of termination for term rewriting systems with ordered sorts, in [*Hamza*, 2005], pp. 674–679, 2005a.

Iwami, M., Persistence of termination for right-linear overlay term rewriting systems, in [*Hamza*, 2005], pp. 686–691, 2005b.

Kirchner, C., H. Kirchner, and A. S. de Oliveira, Analysis of Rewrite-Based Access Control Policies, in *Proc. 3rd Int'l Workshop on Security and Rewriting Techniques, SecreT 2008*, Elsevier ENTCS, 2008.

Knuth, D. E., and P. B. Bendix, Simple word problems in universal algebras, in *Computational Problems in Abstract Algebra*, pp. 263–297, 1970.

Lassez, J.-L., M. J. Maher, and K. Marriott, Unification Revisited, in *Foundations of Deductive Databases and Logic Programming*, edited by J. Minker, pp. 587–625, Morgan Kaufmann, Los Altos, Ca., 1988.

López-Fraguas, F., and J. Sánchez-Hernández, TOY: A Multiparadigm Declarative System, in *Proc. of RTA'99*, *LNCS*, vol. 1631, pp. 244–247, Springer, 1999.

Marche, C., and H. Zantema, The termination competition, in *Term Rewriting and Applications*, vol. 4533, pp. 303–313, Springer, 2007.

Marchiori, M., Proving existential termination of normal logic programs, in *AMAST*, *Lecture Notes in Computer Science*, vol. 1101, edited by M. Wirsing and M. Nivat, pp. 375–390, Springer, 1996.

Martelli, A., C. Moiso, and G. F. Rossi, An Algorithm for Unification in Equational Theories, in *IEEE Symposium on Logic Programming*, pp. 180–186, IEEE Computer Society Press, Los Alamitos, CA, USA, 1986.

Meseguer, J., Multiparadigm logic programming, in *3rd Int'l Conference on Algebraic and Logic Programming, ALP'92*, *LNCS*, vol. 632, pp. 158–200, Springer-Verlag, Berlin, 1992.

Meseguer, J., and P. Thati, Symbolic reachability analysis using narrowing and its application to verification of cryptographic protocols, *Higher-Order and Symbolic Computation*, *20*(1-2), 123–160, 2007.

Middeldorp, A., Approximations for strategies and termination, *Electr. Notes Theor. Comput. Sci.*, *70*(6), 2002.

Middeldorp, A., and E. Hamoen, Completeness Results for Basic Narrowing, *Journal of Applicable Algebra in Engineering, Communication and Computing*, *5*, 313–353, 1994.

Middeldorp, A., S. Okui, and T. Ida, Lazy narrowing: Strong completeness and eager variable elimination, *Theoretical Computer Science*, *167*(1,2), 95–130, 1996.

Mitra, S., and N. Dershowitz, Matching and Unification in Rewrite Theories, 1996.

Narradar, The Narradar Termination Tool, http://safe-tools.dsic.upv.es/narradar.

Newman, M. H. A., On theories with a combinatorial definition of equivalence, *Annals of Mathematics*, *43(2)*, 223–243, 1942.

Nguyen, M. T., P. Schneider-Kamp, D. de Schreye, and J. Giesl, Termination Analysis of Logic Programs based on Dependency Graphs, in *17th Int'l Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR 2007*, *LNCS*, vol. 4915, pp. 8–22, Springer, 2008.

Nguyen, M. T., D. D. Schreye, J. Giesl, and P. Schneider-Kamp, Polytool: polynomial interpretations as a basis for termination analysis of logic programs, *CoRR*, *abs/0912.4360*, 2009.

Nieuwenhuis, R., Basic Paramodulation and Decidable Theories, in *Eleventh Annual IEEE Symposium On Logic In Computer Science (LICS)*, pp. 473–483, IEEE Computer Society Press, Los Alamitos, CA, USA, 1996.

Nishida, N., and K. Miura, Dependency graph method for proving termination of narrowing, in *8th Int'l Workshop on Termination, WST'06*, pp. 12–16, 2006.

Nishida, N., and G. Vidal, Termination of Narrowing via Termination of Rewriting, *Applicable Algebra in Engineering, Communications and Computing*, *21*(3), 177–225, 2010.

Nishida, N., M. Sakai, and T. Sakabe, Narrowing-based simulation of term rewriting systems with extra variables., *Electr. Notes Theor. Comput. Sci.*, *86*(3), 2003.

Ohlebusch, E., Modular properties of composable term rewriting systems, *Journal of Symbolic Computation*, *20*, 1–41, 1995.

Ohlebusch, E., *Advanced topics in term rewriting*, Springer-Verlag, London, UK, 2002.

Payet, É., Detecting non-termination of term rewriting systems using an unfolding operator, in *LOPSTR*, *Lecture Notes in Computer Science*, vol. 4407, edited by G. Puebla, pp. 194–209, Springer, 2006.

Payet, É., Loop detection in term rewriting using the eliminating unfoldings, *Theor. Comput. Sci.*, *403*(2-3), 307–327, 2008.

Payet, É., and F. Mesnard, Nontermination inference of logic programs, *ACM Trans. Program. Lang. Syst.*, *28*(2), 256–289, 2006.

Peyton Jones, S., *Haskell 98 Language and Libraries: the Revised Report*, Cambridge University Press, 2003.

Plotkin, G. D., A note on inductive generalization, *Machine Intelligence*, *5*, 153–165, 1970.

Prehofer, C., On Modularity in Term Rewriting and Narrowing, in *First Int'l Conference on Constraints in Computational Logic*, *Lecture Notes in Computer Science*, vol. 845, pp. 253–268, Springer, 1994.

Rao, M. K., Modular proofs for completeness of hierarchical term rewriting systems, *Theoretical Computer Science*, 1995a.

Rao, M. R. K. K., Semi-completeness of hierarchical and super-hierarchical combinations of term rewriting systems, in *TAPSOFT '95: Proceedings of the 6th International Joint Conference CAAP/FASE on Theory and Practice of Software Development*, pp. 379–393, Springer-Verlag, London, UK, 1995b.

Reddy, U. S., Narrowing As The Operational Semantics Of Functional Languages, in *IEEE Symposium on Logic Programming*, pp. 138–151, 1985.

Réty, P., Improving Basic Narrowing Techniques and Commutation Properties, *Tech. Rep. RR-0681*, INRIA - Lorraine, available at http://www.inria.fr/rrrt/rr-0681.html, 1987.

Réty, P., Improving Basic Narrowing Techniques, in *2nd Int'l Conference on Rewriting Techniques and Applications (RTA)*, *Lecture Notes in Computer Science*, vol. 256, pp. 228–241, Springer-Verlag, 1987.

Schmidt-Schauß, M., Unification in a Combination of Arbitrary Disjoint Equational Theories, in *9th Int'l Conference on Automated Deduction (CADE)*, *Lecture Notes in Computer Science*, vol. 310, pp. 378–396, Springer-Verlag, 1988.

Schneider-Kamp, P., Static termination analysis for prolog using term rewriting and sat solving, Ph.D. thesis, RWTH Aachen University, 2008.

Schneider-Kamp, P., private communications, 2009.

Schneider-Kamp, P., R. Thiemann, E. Annov, M. Codish, and J. Giesl, Proving termination using recursive path orders and sat solving, in *FroCos*, *LNCS*, vol. 4720, edited by B. Konev and F. Wolter, pp. 267–282, Springer, 2007.

Schneider-Kamp, P., J. Giesl, and M. T. Nguyen, The dependency triple framework for termination of logic programs, in *LOPSTR*, *Lecture Notes in Computer Science*, vol. 6037, edited by D. D. Schreye, pp. 37–51, Springer, 2009a.

Schneider-Kamp, P., J. Giesl, A. Serebrenik, and R. Thiemann, Automated termination proofs for logic programs by term rewriting, *ACM Trans. Comput. Logic*, *11*(1), 1–52, 2009b.

Sheard, T., Type-Level Computation Using Narrowing in Ωmega, in *Programming Languages meets Program Verification*, vol. 1643, 2006.

Siekmann, J. H., Unification Theory, *Journal of Symbolic Computation*, *7*, 207–274, 1989a.

Siekmann, J. H., Unification Theory, *Journal of Symbolic Computation*, *7*(3/4), 207–274, 1989b.

Swiderski, S., M. Parting, J. Giesl, C. Fuhs, and P. Schneider-Kamp, Termination analysis by dependency pairs and inductive theorem proving, in *CADE*, *Lecture Notes in Computer Science*, vol. 5663, edited by R. A. Schmidt, pp. 322–338, Springer, 2009.

TeReSe (Ed.), *Term Rewriting Systems*, Cambridge University Press, Cambridge, UK, 2003.

Thiemann, R., The DP framework for proving termination of term rewriting, Ph.D. thesis, RWTH Aachen University, 2007.

Toyama, Y., Counterexamples to termination for the direct sum of term rewriting systems, *Information Processing Letters*, *25*(3), 141–143, doi:http://dx.doi.org/10.1016/0020-0190(87)90122-0, 1987.

Urbain, X., Modular & incremental automated termination proofs, *Int. J. Approx. Reasoning*, *32*(4), 315–355, 2004.

Vidal, G., Termination of Narrowing in Left-Linear Constructor Systems, in *9th Int'l Symposium on Functional and Logic Languages (FLOPS 2008)*, edited by J. Garrigue and M. Hermenegildo, pp. 113–129, Springer LNCS 4989, 2008.

Yamamoto, A., Completeness of extended unification based on basic narrowing, *qir.kyushu-u.ac.jp*, 1988.

Yices, Yices: an smt solver, http://yices.csl.sri.com/.

Zantema, H., Termination of term rewriting: Interpretation and type elimination, *J. Symb. Comput.*, *17*(1), 23–50, 1994.

Zantema, H., Relative termination in term rewriting, in *7th Int'l Workshop on Termination, WST'04*, pp. 51–55, 2004.

Index