Document downloaded from:

http://hdl.handle.net/10251/202279

This paper must be cited as:

Gutiérrez Gil, R.; Lucas Alba, S. (2010). Proving Termination in the Context-Sensitive Dependency Pair Framework. Lecture Notes in Computer Science. 6381:18-34. https://doi.org/10.1007/978-3-642-16310-4_3

The final publication is available at

http://doi.org/10.1007/978-3-642-16310-4_3

Copyright Springer-Verlag

Additional Information

Proving Termination in the Context-Sensitive Dependency Pair Framework*

Raúl Gutiérrez[1] and Salvador Lucas[1]

ELP Group, DSIC, Universitat Politècnica de València
Camí de Vera s/n, 46022 València, Spain

Abstract. Termination of *context-sensitive rewriting* (CSR) is an interesting problem with several applications in the fields of term rewriting and in the analysis of programming languages like CafeOBJ, Maude, OBJ, etc. The dependency pair approach, one of the most powerful techniques for proving termination of rewriting, has been adapted to be used for proving termination of CSR. The corresponding notion of *context-sensitive dependency pair* (CSDP) is different from the standard one in that *collapsing pairs* (i.e., rules whose right-hand side is a variable) are considered. Although the implementation and practical use of CSDPs lead to a powerful framework for proving termination of CSR, handling collapsing pairs is not easy and often leads to impose heavy requirements over the base orderings which are used to achieve the proofs. A recent proposal removes collapsing pairs by transforming them into sets of new (standard) pairs. In this way, though, the role of collapsing pairs for modeling context-sensitive computations gets lost. This leads to a less intuitive and accurate description of the termination behavior of the system. In this paper, we show how to get the best of the two approaches, thus obtaining a powerful *context-sensitive dependency pair framework* which satisfies all practical and theoretical expectations.

1 Introduction

In Context-Sensitive Rewriting (CSR, [1]), a *replacement map* μ satisfying $\mu(\mathsf{f}) \subseteq \{1,\ldots,\mathsf{ar}(\mathsf{f})\}$ for every function symbol f of arity $\mathsf{ar}(\mathsf{f})$ in the signature $\mathcal{F}$ is used to discriminate the argument positions on which the rewriting steps are allowed. In this way, a terminating behavior of (context-sensitive) computations with Term Rewriting Systems (TRSs) can be obtained. CSR has shown useful to model evaluation strategies in programming languages. In particular, it is an essential ingredient to analyze the *termination behavior* of programs in programming languages (like CafeOBJ, Maude, OBJ, etc.) which implement recent presentations of rewriting logic like the *Generalized Rewrite Theories* [2], see [3–5].

* Partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02-02.

Example 1. Consider the following TRS in [6]:

$$\begin{array}{rlrl}
\mathsf{gt}(\mathsf{0},y) & \to \mathsf{false} & \mathsf{p}(\mathsf{0}) & \to \mathsf{0} \\
\mathsf{gt}(\mathsf{s}(x),\mathsf{0}) & \to \mathsf{true} & \mathsf{p}(\mathsf{s}(x)) & \to x \\
\mathsf{gt}(\mathsf{s}(x),\mathsf{s}(y)) & \to \mathsf{gt}(x,y) & \mathsf{minus}(x,y) & \to \mathsf{if}(\mathsf{gt}(y,\mathsf{0}),\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y)),x) \\
\mathsf{if}(\mathsf{true},x,y) & \to x & \mathsf{div}(\mathsf{0},\mathsf{s}(y)) & \to \mathsf{0} \\
\mathsf{if}(\mathsf{false},x,y) & \to y & \mathsf{div}(\mathsf{s}(x),\mathsf{s}(y)) & \to \mathsf{s}(\mathsf{div}(\mathsf{minus}(x,y),\mathsf{s}(y)))
\end{array}$$

with $\mu(\mathsf{if}) = \{1\}$ *and* $\mu(\mathsf{f}) = \{1,\ldots,\mathsf{ar}(\mathsf{f})\}$ *for all other symbols* f*. Note that, if no replacement restriction is considered, then the following sequence is possible and the system would be nonterminating:*

$$\underline{\mathsf{minus}(\mathsf{0},\mathsf{0})} \to^*_{\mathcal{R}} \mathsf{if}(\mathsf{gt}(\mathsf{0},\mathsf{0}),\underline{\mathsf{minus}(\mathsf{0},\mathsf{0})},\mathsf{0}) \to^*_{\mathcal{R}} \ldots,\underline{\mathsf{minus}(\mathsf{0},\mathsf{0})},\ldots \to^*_{\mathcal{R}} \cdots$$

In CSR*, though, this sequence is not possible because reductions on the second argument of the* if*-operator are disallowed due to* $\mu(\mathsf{if}) = \{1\}$*.*

In [7], Arts and Giesl's dependency pair approach [8], a powerful technique for proving termination of rewriting, was adapted to CSR (see [9] for a more recent presentation). Regarding proofs of termination of rewriting, the dependency pair technique focuses on the following idea: since a TRS $\mathcal{R}$ is terminating if there is no infinite rewrite sequence starting from any term, the rules that are really able to produce such infinite sequences are those rules $\ell \to r$ such that r contains some *defined* symbol[1] g. Intuitively, we can think of these rules as representing possible (direct or indirect) recursive calls. Recursion paths associated to each rule $\ell \to r$ are represented as new rules $u \to v$ (called *dependency pairs*) where $u = \mathsf{f}^\sharp(\ell_1,\ldots,\ell_k)$ if $\ell = \mathsf{f}(\ell_1,\ldots,\ell_k)$ and $v = \mathsf{g}^\sharp(s_1,\ldots,s_m)$ if $s = \mathsf{g}(s_1,\ldots,s_m)$ is a subterm of r and g is a defined symbol. The notation $\mathsf{f}^\sharp$ for a given symbol f means that f is *marked*. In practice, we often capitalize f and use F instead of $\mathsf{f}^\sharp$ in our examples. For this reason, the dependency pair technique starts by considering a new TRS $\mathsf{DP}(\mathcal{R})$ which contains all these dependency pairs for each $\ell \to r \in \mathcal{R}$. The rules in $\mathcal{R}$ and $\mathsf{DP}(\mathcal{R})$ determine the so-called *dependency chains* whose finiteness characterizes termination of $\mathcal{R}$ [8]. Furthermore, the dependency pairs can be presented as a *dependency graph*, where the infinite chains are captured by the *cycles* in the graph.

These intuitions are valid for CSR, but the subterms s of the right-hand sides r of the rules $\ell \to r$ which are considered to build the *context-sensitive dependency pairs* $\ell^\sharp \to s^\sharp$ must be μ*-replacing* terms. In sharp contrast with the dependency pair approach, though, we also need *collapsing dependency pairs* $u \to x$ where u is obtained from the left-hand side ℓ of a rule $\ell \to r$ in the usual way, i.e., $u = \ell^\sharp$ but x is a *migrating variable* which is μ-replacing in r but which only occurs at *non-μ-replacing positions* in ℓ [7, 9]. Collapsing pairs are essential in our approach. They express that infinite context-sensitive rewrite sequences can involve not only the kind of recursion which is represented by the *usual* dependency pairs but also a new kind of recursion which is *hidden* inside

[1] A symbol $\mathsf{g} \in \mathcal{F}$ is defined in $\mathcal{R}$ if there is a rule $\ell \to r$ in $\mathcal{R}$ whose left-hand side ℓ is of the form $\mathsf{g}(\ell_1,\ldots,\ell_k)$ for some $k \geq 0$.

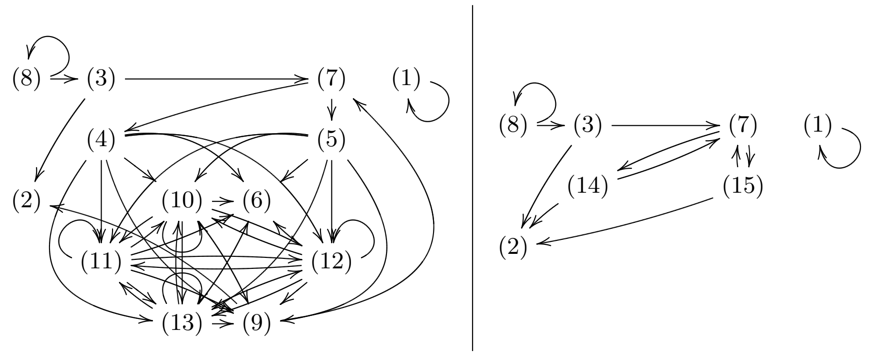

Fig. 1. Dependency graph for Example 1 following [6] (left) and [9] (right)

the non-μ-replacing parts of the terms involved in the infinite sequence until a *migrating* variable within a rule $\ell \to r$ shows them up.

In [6], a transformation that replaces the *collapsing pairs* by a new set of pairs that simulate their behavior was introduced. This new set of pairs is used to simplify the definition of context-sensitive dependency chain; but, on the other hand, we loose the intuition of what collapsing pairs mean in a context-sensitive rewriting chain. And understanding the new dependency graph is harder.

Example 2. (Continuing Example 1) If we follow the transformational definition in [6], we have the following dependency pairs (a new symbol U *is introduced):*

$$\mathsf{GT}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{GT}(x,y) \quad (1)$$
$$\mathsf{M}(x,y) \to \mathsf{GT}(y,0) \quad (2)$$
$$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y) \quad (3)$$
$$\mathsf{IF}(\mathsf{true},x,y) \to \mathsf{U}(x) \quad (4)$$
$$\mathsf{IF}(\mathsf{false},x,y) \to \mathsf{U}(y) \quad (5)$$
$$\mathsf{U}(\mathsf{p}(x)) \to \mathsf{P}(x) \quad (6)$$
$$\mathsf{M}(x,y) \to \mathsf{IF}(\mathsf{gt}(y,0),\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y)),x) \quad (7)$$
$$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x,y),\mathsf{s}(y)) \quad (8)$$
$$\mathsf{U}(\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y))) \to \mathsf{M}(\mathsf{p}(x),\mathsf{p}(y)) \quad (9)$$
$$\mathsf{U}(\mathsf{p}(x)) \to \mathsf{U}(x) \quad (10)$$
$$\mathsf{U}(\mathsf{p}(y)) \to \mathsf{U}(y) \quad (11)$$
$$\mathsf{U}(\mathsf{minus}(x,y)) \to \mathsf{U}(x) \quad (12)$$
$$\mathsf{U}(\mathsf{minus}(x,y)) \to \mathsf{U}(y) \quad (13)$$

and the dependency graph has the unreadable aspect shown in Figure 1 (left). In contrast, if we consider the original definition of CSDPs and CSDG in [7, 9], our set of dependency pairs is the following:

$$\mathsf{GT}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{GT}(x,y) \quad (1)$$
$$\mathsf{M}(x,y) \to \mathsf{GT}(y,0) \quad (2)$$
$$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{M}(x,y) \quad (3)$$
$$\mathsf{M}(x,y) \to \mathsf{IF}(\mathsf{gt}(y,0),\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y)),x) \quad (7)$$
$$\mathsf{D}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{D}(\mathsf{minus}(x,y),\mathsf{s}(y)) \quad (8)$$
$$\mathsf{IF}(\mathsf{true},x,y) \to x \quad (14)$$
$$\mathsf{IF}(\mathsf{false},x,y) \to y \quad (15)$$

and the dependency graph is much more clear, see Figure 1 (right).

The work in [6] was motivated by the fact that mechanizing proofs of termination of CSR according to the results in [7] can be difficult due to the presence of collapsing dependency pairs. The problem is that [7] imposes hard restrictions on the orderings which are used in proofs of termination of CSR when collapsing dependency pairs are present. In this paper we address this problem in a different way. We keep collapsing CSDPs (and their descriptive power and simplicity) while the practical problems for handling them are overcome.

After some preliminaries in Section 2, in Section 3 we introduce the notion of *hidden term* and *hiding context* and discuss their role in infinite μ-rewrite sequences. In Section 4 we introduce a new notion of CSDP chain which is well-suited for mechanizing proofs of termination of CSR with CSDPs. In Section 5 we introduce our dependency pair framework for proving termination of CSR. Furthermore, we show that with the new definition we can also use all the existing processors from the two previous approaches and we can define new powerful processors. Section 6 shows an specific example of the power of this framework. Section 7 shows our experimental results. Section 8 discusses the differences between our approach and the one in [6]. Section 9 concludes. Proofs can be found in [10].

2 Preliminaries

We assume a basic knowledge about standard definitions and notations for term rewriting as given in, e.g., [11]. Positions $p, q, \ldots$ are represented by chains of positive natural numbers used to address subterms of t. Given positions p, q, we denote its concatenation as $p.q$. If p is a position, and Q is a set of positions, then $p.Q = \{p.q \mid q \in Q\}$. We denote the root or top position by Λ. The set of positions of a term t is $\mathcal{P}os(t)$. Positions of nonvariable symbols $\mathsf{f} \in \mathcal{F}$ in $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ are denoted as $\mathcal{P}os_{\mathcal{F}}(t)$. The *subterm* at position p of t is denoted as $t|_p$ and $t[s]_p$ is the term t with the subterm at position p replaced by s. We write $t \trianglerighteq s$ if $s = t|_p$ for some $p \in \mathcal{P}os(t)$ and $t \triangleright s$ if $t \trianglerighteq s$ and $t \neq s$. The symbol labeling the root of t is denoted as $\mathsf{root}(t)$. A *substitution* is a mapping $\sigma : \mathcal{X} \to \mathcal{T}(\mathcal{F}, \mathcal{X})$ from a set of variables $\mathcal{X}$ into the set $\mathcal{T}(\mathcal{F}, \mathcal{X})$ of terms built from the symbols in the *signature* $\mathcal{F}$ and the variables in $\mathcal{X}$. A *context* is a term $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{X})$ with a 'hole' $\Box$ (a fresh constant symbol). A *rewrite rule* is an ordered pair (ℓ, r), written $\ell \to r$, with $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$, $\ell \notin \mathcal{X}$ and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(\ell)$. The left-hand side (*lhs*) of the rule is ℓ and r is the right-hand side (*rhs*). A *TRS* is a pair $\mathcal{R} = (\mathcal{F}, R)$ where $\mathcal{F}$ is a signature and R is a set of rewrite rules over terms in $\mathcal{T}(\mathcal{F}, \mathcal{X})$. Given $\mathcal{R} = (\mathcal{F}, R)$, we consider $\mathcal{F}$ as the disjoint union $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$ of symbols $\mathsf{c} \in \mathcal{C}$, called *constructors* and symbols $\mathsf{f} \in \mathcal{D}$, called *defined symbols*, where $\mathcal{D} = \{\mathsf{root}(\ell) \mid \ell \to r \in R\}$ and $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$.

In the following, we introduce some notions and notation about CSR [1]. A mapping $\mu : \mathcal{F} \to \wp(\mathbb{N})$ is a *replacement map* if $\forall \mathsf{f} \in \mathcal{F},\ \mu(\mathsf{f}) \subseteq \{1, \ldots, \mathsf{ar}(\mathsf{f})\}$. Let $M_{\mathcal{F}}$ be the set of all replacement maps (or $M_{\mathcal{R}}$ for the replacement maps of a TRS $\mathcal{R} = (\mathcal{F}, R)$). The set of *$\mu$-replacing positions* $\mathcal{P}os^{\mu}(t)$ of $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\Lambda\}$, if $t \in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t|_i)$, if $t \notin \mathcal{X}$. The set of *μ-replacing variables* of t is $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$ and $\mathcal{V}ar^{\not\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os(t) \setminus \mathcal{P}os^{\mu}(t), t|_p = x\}$ is the set of *non-μ-replacing variables* of t. Note that $\mathcal{V}ar^{\mu}(t)$ and $\mathcal{V}ar^{\not\mu}(t)$ do not need to be disjoint. The *μ-replacing subterm* relation $\trianglerighteq_{\mu}$ is given by $t \trianglerighteq_{\mu} s$ if there is $p \in \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. We write $t \triangleright_{\mu} s$ if $t \trianglerighteq_{\mu} s$ and $t \neq s$. We write

$t \rhd_{\not\mu} s$ to denote that s is a *non-μ-replacing strict subterm* of t, i.e., there is a *non-μ-replacing position* $p \in \mathcal{P}os(t) \setminus \mathcal{P}os^{\mu}(t)$ such that $s = t|_p$. In CSR, we (only) contract *μ-replacing redexes*: t μ-rewrites to s, written $t \hookrightarrow_{\mathcal{R},\mu} s$ (or $t \overset{p}{\hookrightarrow}_{\mathcal{R},\mu} s$ to make position p explicit), iff there are $\ell \to r \in R$, $p \in \mathcal{P}os^{\mu}(t)$ and a substitution σ such that $t|_p = \sigma(\ell)$ and $s = t[\sigma(r)]_p$; $t \overset{>q}{\hookrightarrow}_{\mathcal{R},\mu} s$ means that the μ-rewrite step is applied below position q, i.e., $p > q$. We say that a variable x is *migrating* in $\ell \to r \in R$ if $x \in \mathcal{V}ar^{\mu}(r) \setminus \mathcal{V}ar^{\mu}(\ell)$. A term t is *μ-terminating* if there is no infinite μ-rewrite sequence $t = t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \cdots \hookrightarrow_{\mathcal{R},\mu} t_n \hookrightarrow_{\mathcal{R},\mu} \cdots$ starting from t. A TRS $\mathcal{R} = (\mathcal{F}, R)$ is *μ-terminating* if $\hookrightarrow_{\mathcal{R},\mu}$ is terminating. A pair $(\mathcal{R}, \mu)$ where $\mathcal{R}$ is a TRS and $\mu \in M_{\mathcal{R}}$ is often called a *CS-TRS*.

3 Infinite μ-Rewrite Sequences

Let $\mathcal{M}_{\infty,\mu}$ be a set of *minimal non-μ-terminating terms* in the following sense: t belongs to $\mathcal{M}_{\infty,\mu}$ if t is non-μ-terminating and every strict *μ-replacing* subterm s of t (i.e., $t \rhd_{\mu} s$) is μ-terminating [7]. Minimal terms allow us to characterize infinite μ-rewrite sequences [9]. In [9], we show that if we have migrating variables x that "unhide" infinite computations starting from terms u which are introduced by the binding $\sigma(x)$ of the variable, then we can obtain information about the "incoming" term u if this term does not occur in the initial term of the sequence. In order to formalize this, we need a restricted notion of minimality.

Definition 1 (Strongly Minimal Terms [9]). *Let $\mathcal{T}_{\infty,\mu}$ be a set of* strongly minimal non-μ-terminating terms *in the following sense: t belongs to $\mathcal{T}_{\infty,\mu}$ if t is non-μ-terminating and every strict subterm u (i.e., $t \rhd u$) is μ-terminating. It is obvious that* $\mathsf{root}(t) \in \mathcal{D}$ *for all $t \in \mathcal{T}_{\infty,\mu}$.*

Every non-μ-terminating term has a subterm that is strongly minimal. Then, given a non-μ-terminating term t we can always find a subterm $t_0 \in \mathcal{T}_{\infty,\mu}$ of t which starts a *minimal infinite μ-rewrite sequence* of the form $t_0 \overset{>\Lambda}{\hookrightarrow}{}^{*}_{\mathcal{R},\mu} \sigma_1(\ell_1) \overset{\Lambda}{\hookrightarrow}_{\mathcal{R},\mu} \sigma_1(r_1) \trianglerighteq_{\mu} t_1 \overset{>\Lambda}{\hookrightarrow}{}^{*}_{\mathcal{R},\mu} \sigma_2(\ell_2) \overset{\Lambda}{\hookrightarrow}_{\mathcal{R},\mu} \sigma_2(r_2) \trianglerighteq_{\mu} t_2 \overset{>\Lambda}{\hookrightarrow}{}^{*}_{\mathcal{R},\mu} \cdots$ where $t_i, \sigma_i(\ell_i) \in \mathcal{M}_{\infty,\mu}$ for all $i > 0$ [9]. Theorem 1 below tells us that we have two possibilities:

- The minimal non-μ-terminating terms $t_i \in \mathcal{M}_{\infty,\mu}$ in the sequence are partially introduced by a μ-replacing nonvariable subterm of the right-hand sides r_i of the rules $\ell_i \to r_i$.
- The minimal non-μ-terminating terms $t_i \in \mathcal{M}_{\infty,\mu}$ in the sequence are introduced by instantiated *migrating variables* x_i of (the respective) rules $\ell_i \to r_i$, i.e., $x_i \in \mathcal{V}ar^{\mu}(r_i) \setminus \mathcal{V}ar^{\mu}(\ell_i)$. Then, t_i is partially introduced by terms occurring at non-μ-replacing positions in the right-hand sides of the rules (*hidden terms*) within a given (*hiding*) context.

We use the following functions [7, 9]: $\textsc{Ren}^{\mu}(t)$, which *independently* renames all *occurrences* of μ-replacing variables by using new fresh variables which are not in $\mathcal{V}ar(t)$, and $\textsc{Narr}^{\mu}_{\mathcal{R}}(t)$, which indicates whether t is μ-narrowable[2] (w.r.t. the intended TRS $\mathcal{R}$).

A nonvariable term $t \in \mathcal{T}(\mathcal{F},\mathcal{X}) \setminus \mathcal{X}$ is a *hidden term* [6, 9] if there is a rule $\ell \to r \in R$ such that t is a non-μ-replacing subterm of r. In the following, $\mathcal{HT}(\mathcal{R},\mu)$ is the set of all hidden terms in $(\mathcal{R},\mu)$ and $\mathcal{NHT}(\mathcal{R},\mu)$ the set of μ-narrowable hidden terms headed by a defined symbol:

$$\mathcal{NHT}(\mathcal{R},\mu) = \{t \in \mathcal{HT}(\mathcal{R},\mu) \mid \mathsf{root}(t) \in \mathcal{D} \text{ and } \textsc{Narr}^{\mu}_{\mathcal{R}}(\textsc{Ren}^{\mu}(t))\}$$

Definition 2 (Hiding Context). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. A function symbol* f hides position i in the rule $\ell \to r \in \mathcal{R}$ *if $r \rhd_{\not\mu} \mathsf{f}(r_1,\ldots,r_n)$ for some terms $r_1,\ldots,r_n$, and there is $i \in \mu(\mathsf{f})$ such that r_i contains a μ-replacing defined symbol (i.e., $\mathcal{P}os^{\mu}_{\mathcal{D}}(r_i) \neq \varnothing$) or a variable $x \in (\mathcal{V}ar^{\not\mu}(\ell) \cap \mathcal{V}ar^{\not\mu}(r)) \setminus (\mathcal{V}ar^{\mu}(\ell) \cup \mathcal{V}ar^{\mu}(r))$ which is μ-replacing in r_i (i.e., $x \in \mathcal{V}ar^{\mu}(r_i)$). A context $C[\Box]$ is* hiding *[6] if $C[\Box] = \Box$, or $C[\Box] = \mathsf{f}(t_1,\ldots,t_{i-1},C'[\Box],t_{i+1},\ldots,t_k)$, where* f *hides position i and $C'[\Box]$ is a hiding context.*

Definition 2 is a refinement of [6, Definition 7], where the new condition $x \in (\mathcal{V}ar^{\not\mu}(\ell) \cap \mathcal{V}ar^{\not\mu}(r)) \setminus (\mathcal{V}ar^{\mu}(\ell) \cup \mathcal{V}ar^{\mu}(r))$ is useful to discard contexts that are not valid when minimality is considered.

Example 3. The hidden terms in Example 1 are $\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y))$, $\mathsf{p}(x)$ and $\mathsf{p}(y)$. Symbol minus hides positions 1 and 2, but p hides no position. Without the new condition in Definition 2, p would hide position 1.

These notions are used and combined to model infinite context-sensitive rewrite sequences starting from strongly minimal non-μ-terminating terms as follows.

Theorem 1 (Minimal Sequence). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. For all $t \in \mathcal{T}_{\infty,\mu}$, there is an infinite sequence*

$$t = t_0 \xhookrightarrow{>\Lambda}{}^{*}_{\mathcal{R},\mu} \sigma_1(\ell_1) \xhookrightarrow{\Lambda}_{\mathcal{R},\mu} \sigma_1(r_1) \trianglerighteq_{\mu} t_1 \xhookrightarrow{>\Lambda}{}^{*}_{\mathcal{R},\mu} \sigma_2(\ell_2) \xhookrightarrow{\Lambda}_{\mathcal{R},\mu} \cdots$$

where, for all $i \geq 1$, $\ell_i \to r_i \in R$ are rewrite rules, σ_i are substitutions, and terms $t_i \in \mathcal{M}_{\infty,\mu}$ are minimal non-μ-terminating terms such that either

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

4 Chains of Context-Sensitive Dependency Pairs

In this section, we revise the definition of chain of context-sensitive dependency pairs given in [9]. First, we recall the notion of context-sensitive dependency pair.

[2] A term s *μ-narrows* to the term t if there is a nonvariable position $p \in \mathcal{P}os^{\mu}_{\mathcal{F}}(s)$ and a rule $\ell \to r$ such that $s|_p$ and ℓ *unify* with *mgu* σ, and $t = \sigma(s[r]_p)$.

Definition 3 (Context-Sensitive Dependency Pairs [9]). *Let $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$ be a TRS and $\mu \in M_{\mathcal{F}}$. We define $\mathsf{DP}(\mathcal{R},\mu) = \mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu) \cup \mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu)$ to be set of* context-sensitive dependency pairs *(CSDPs) where:*

$$\mathsf{DP}_{\mathcal{F}}(\mathcal{R},\mu) = \{\ell^\sharp \to s^\sharp \mid \ell \to r \in R, r \trianglerighteq_\mu s, \mathsf{root}(s) \in \mathcal{D}, \ell \ntriangleright_\mu s, \textsc{Narr}^\mu_{\mathcal{R}}(\textsc{Ren}^\mu(s))\}$$
$$\mathsf{DP}_{\mathcal{X}}(\mathcal{R},\mu) = \{\ell^\sharp \to x \mid \ell \to r \in R, x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(\ell)\}$$

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

Now, we provide a new notion of *chain* of CSDPs. In contrast to [6], we store the information about hidden terms and hiding contexts which is relevant to model infinite minimal μ-rewrite sequences as a new *unhiding TRS* instead of introducing them as new (transformed) pairs.

Definition 4 (Unhiding TRS). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. We define $\mathsf{unh}(\mathcal{R},\mu)$ as the TRS consisting of the following rules:*

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

Example 4. The unhiding TRS $\mathsf{unh}(\mathcal{R},\mu)$ for $\mathcal{R}$ and μ in Example 1 is:

$$\mathsf{minus}(\mathsf{p}(x),\mathsf{p}(y)) \to \mathsf{M}(\mathsf{p}(x),\mathsf{p}(y)) \quad (16) \qquad \mathsf{minus}(x,y) \to y \quad (18)$$
$$\mathsf{p}(x) \to \mathsf{P}(x) \quad (17) \qquad \mathsf{minus}(x,y) \to x \quad (19)$$

Definitions 3 and 4 lead to a suitable notion of *chain* which captures minimal infinite μ-rewrite sequences according to the description in Theorem 1. In the following, given a TRS $\mathcal{S}$, we let $\mathcal{S}_{\triangleright_\mu}$ be the rules from $\mathcal{S}$ of the form $s \to t \in \mathcal{S}$ and $s \triangleright_\mu t$; and $\mathcal{S}_\sharp = \mathcal{S} \setminus \mathcal{S}_{\triangleright_\mu}$.

Definition 5 (Chain of Pairs - Minimal Chain). *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$. A $(\mathcal{P},\mathcal{R},\mathcal{S},\mu)$-chain is a finite or infinite sequence of pairs $u_i \to v_i \in \mathcal{P}$, together with a substitution σ satisfying that, for all $i \geq 1$,*

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

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

Notice that if rules $\mathsf{f}(x_1,\ldots,x_k) \to x_i$ for all $\mathsf{f} \in \mathcal{D}$ and $i \in \mu(\mathsf{f})$ (where $x_1,\ldots,x_k$ are variables) are used in Item 1 of Definition 4, then Definition 5 yields the notion of chain in [9]; and if, additionally, rules $\mathsf{f}(x_1,\ldots,x_k) \to \mathsf{f}^\sharp(x_1,\ldots,x_k)$ for all $\mathsf{f} \in \mathcal{D}$ are used in Item 2 of Definition 4, then we have the original notion of chain in [7]. Thus, the new definition covers all previous ones.

Theorem 2 (Soundness and Completeness of CSDPs). *Let $\mathcal{R}$ be a TRS and $\mu \in M_{\mathcal{R}}$. A CS-TRS $(\mathcal{R},\mu)$ is terminating if and only if there is no infinite $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$-chain.*

5 Context-Sensitive Dependency Pair Framework

In the DP framework [12], proofs of termination are handled as *termination problems* involving two TRSs $\mathcal{P}$ and $\mathcal{R}$ instead of just the 'target' TRS $\mathcal{R}$. In our setting we start with the following definition (see also [6, 9]).

Definition 6 (CS Problem and CS Processor). *A* CS problem τ *is a tuple* $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$*, where* $\mathcal{R}$*,* $\mathcal{P}$ *and* $\mathcal{S}$ *are TRSs, and* $\mu \in M_{\mathcal{R}\cup\mathcal{P}\cup\mathcal{S}}$*. The* CS problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ *is* finite *if there is no infinite* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$*-chain. The* CS problem $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ *is* infinite *if* $\mathcal{R}$ *is non-*μ*-terminating or there is an infinite minimal* $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$*-chain.*

A CS processor Proc *is a mapping from CS problems into sets of CS problems. Alternatively, it can also return "*no*". A CS processor* Proc *is* sound *if for all CS problems* τ*,* τ *is finite whenever* $\mathsf{Proc}(\tau) \neq \mathsf{no}$ *and* $\forall \tau' \in \mathsf{Proc}(\tau)$*,* τ' *is finite. A CS processor* Proc *is* complete *if for all CS problems* τ*,* τ *is infinite whenever* $\mathsf{Proc}(\tau) = \mathsf{no}$ *or* $\exists \tau' \in \mathsf{Proc}(\tau)$ *such that* τ' *is infinite.*

In order to prove the μ-termination of a TRS $\mathcal{R}$, we adapt the result from [12] to CSR.

Theorem 3 (CSDP Framework). *Let* $\mathcal{R}$ *be a TRS and* $\mu \in M_{\mathcal{R}}$*. We construct a tree whose nodes are labeled with CS problems or* "yes" *or* "no"*, and whose root is labeled with* $(\mathsf{DP}(\mathcal{R}, \mu), \mathcal{R}, \mathsf{unh}(\mathcal{R}, \mu), \mu^\sharp)$*. For every inner node labeled with* τ*, there is a sound processor* Proc *satisfying one of the following conditions:*

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

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

In the following subsections we describe a number of sound and complete CS processors.

5.1 Collapsing Pair Processors

The following processor integrates the transformation of [6] into our framework. The pairs in a CS-TRS $(\mathcal{P}, \mu)$, where $\mathcal{P} = (\mathcal{G}, P)$, are partitioned as follows: $P_{\mathcal{X}} = \{u \to v \in P \mid v \in \mathcal{V}ar(u) \setminus \mathcal{V}ar^{\mu}(u)\}$ and $P_{\mathcal{G}} = P \setminus P_{\mathcal{X}}$.

Theorem 4 (Collapsing Pair Transformation). *Let* $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ *be a CS problem where* $\mathcal{P} = (\mathcal{G}, P)$ *and* P_{U} *be given by the following rules:*

- $u \to \mathsf{U}(x)$ *for every* $u \to x \in \mathcal{P}_{\mathcal{X}}$*,*
- $\mathsf{U}(s) \to \mathsf{U}(t)$ *for every* $s \to t \in \mathcal{S}_{\rhd_\mu}$*, and*

- *$\mathsf{U}(s) \to t$ for every $s \to t \in \mathcal{S}_\sharp$.*

Here, U is a new fresh symbol. Let $\mathcal{P}' = (\mathcal{G} \cup \{\mathsf{U}\}, P')$ where $P' = (P \setminus P_\mathcal{X}) \cup P_\mathsf{U}$, and μ' extends μ by $\mu'(\mathsf{U}) = \varnothing$. The processor Proc_{eColl} given by $\mathsf{Proc}_{eColl}(\tau) = \{(\mathcal{P}', \mathcal{R}, \varnothing, \mu')\}$ is sound and complete.

Now, we can apply all CS processors from [6] and [9] which did not consider any $\mathcal{S}$ component in CS problems.

In our framework, we can also apply specific processors for collapsing pairs that are very useful, but these only are used if we have collapsing pairs in the chains (as in [9]). For instance, we can use the processor in Theorem 5 below, which is often applied in proofs of termination of CSR with MU-TERM [13, 14]. The subTRS of $\mathcal{P}_\mathcal{X}$ containing the rules whose migrating variables occur on non-μ-replacing immediate subterms in the left-hand side is $\mathcal{P}^1_\mathcal{X} = \{\mathsf{f}(u_1, \ldots, u_k) \to x \in \mathcal{P}_\mathcal{X} \mid \exists i, 1 \leq i \leq k, i \notin \mu(\mathsf{f}), x \in \mathcal{V}ar(u_i)\}$.

Theorem 5 (Basic CS Processor for Collapsing Pairs). *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem where $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ and $\mathcal{S} = (\mathcal{H}, S)$. Assume that (1) all the rules in $\mathcal{S}_\sharp$ are noncollapsing, i.e., for all $s \to t \in \mathcal{S}_\sharp$, $t \notin \mathcal{X}$ (2) $\{\mathsf{root}(t) \mid s \to t \in \mathcal{S}_\sharp\} \cap \mathcal{D} = \varnothing$ and (3) for all $s \to t \in \mathcal{S}_\sharp$, we have that $s = \mathsf{f}(s_1, \ldots, s_k)$ and $t = \mathsf{g}(s_1, \ldots, s_k)$ for some $k \in \mathbb{N}$, funtion symbols $\mathsf{f}, \mathsf{g} \in \mathcal{H}$, and terms $s_1, \ldots, s_k$. Then, the processors Proc_{Coll1} given by*

$$\mathsf{Proc}_{Coll1}(\tau) = \begin{cases} \varnothing & \textit{if } \mathcal{P} = \mathcal{P}^1_\mathcal{X} \textit{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \textit{otherwise} \end{cases}$$

is sound and complete.

Example 5. *(Continuing Example 1) Consider the CS problem $\tau = (\mathcal{P}_4, \mathcal{R}, \mathcal{S}_3, \mu^\sharp)$ where $\mathcal{P}_4 = \{(14), (15)\}$ and $\mathcal{S}_3 = \{(16), (18), (19)\}$. We can apply $\mathsf{Proc}_{Coll1}(\tau)$ to conclude that the CS problem τ is finite.*

5.2 Context-Sensitive Dependency Graph

In the DP-approach [8, 12], a *dependency graph* is associated to the TRS $\mathcal{R}$. The nodes of the graph are the dependency pairs in $\mathsf{DP}(\mathcal{R})$ and there is an arc from a dependency pair $u \to v$ to a dependency pair $u' \to v'$ if there are substitutions θ and θ' such that $\theta(v) \to^*_\mathcal{R} \theta'(u')$. In our setting, we have the following.

Definition 7 (Context-Sensitive Graph of Pairs). *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$. The* context-sensitive (CS) graph *$\mathsf{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ has $\mathcal{P}$ as the set of nodes. Given $u \to v, u' \to v' \in \mathcal{P}$, there is an arc from $u \to v$ to $u' \to v'$ if $u \to v, u' \to v'$ is a minimal $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$-chain for some substitution σ.*

In termination proofs, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [15]. The following result formalizes the use of SCCs for dealing with CS problems.

Theorem 6 (SCC Processor). *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem. Then, the processor Proc_{SCC} given by*

$$\mathsf{Proc}_{SCC}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) = \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}_\mathcal{Q}, \mu) \mid \mathcal{Q} \textit{ are the pairs of an SCC in } \mathsf{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\}$$

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

The CS graph is not computable. Thus, we have to use an over-approximation of it. In the following definition, we use the function $\text{TCAP}^\mu_\mathcal{R}(t)$, which renames all subterms headed by a 'defined' symbol in $\mathcal{R}$ by new fresh variables if it can be rewritten:

Definition 8 ($\text{TCAP}^\mu_\mathcal{R}$ [9]). *Given a TRS $\mathcal{R}$ and a replacement map μ, we let $\text{TCAP}^\mu_\mathcal{R}$ be as follows:*

$$\text{TCAP}^\mu_\mathcal{R}(x) = y \quad \textit{if } x \textit{ is a variable, and}$$

$$\text{TCAP}^\mu_\mathcal{R}(\mathsf{f}(t_1, \ldots, t_k)) = \begin{cases} \mathsf{f}([t_1]^\mathsf{f}_1, \ldots, [t_k]^\mathsf{f}_k) & \textit{if } \mathsf{f}([t_1]^\mathsf{f}_1, \ldots, [t_k]^\mathsf{f}_k) \textit{ does not unify} \\ & \textit{with } \ell \textit{ for any } \ell \to r \textit{ in } \mathcal{R} \\ y & \textit{otherwise} \end{cases}$$

where y is a new fresh variable, $[s]^\mathsf{f}_i = \text{TCAP}^\mu_\mathcal{R}(s)$ if $i \in \mu(\mathsf{f})$ and $[s]^\mathsf{f}_i = s$ if $i \notin \mu(\mathsf{f})$. We assume that ℓ shares no variable with $\mathsf{f}([t_1]^\mathsf{f}_1, \ldots, [t_k]^\mathsf{f}_k)$ when the unification is attempted.

Definition 9 (Estimated CS Graph of Pairs). *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem. The* estimated CS graph *associated to $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ (denoted $\mathsf{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$) has $\mathcal{P}$ as the set of nodes and arcs which connect them as follows:*

1. *there is an arc from $u \to v \in \mathcal{P}_\mathcal{G}$ to $u' \to v' \in \mathcal{P}$ if $\text{TCAP}^\mu_\mathcal{R}(v)$ and u' unify, and*
2. *there is an arc from $u \to v \in \mathcal{P}_\mathcal{X}$ to $u' \to v' \in \mathcal{P}$ if there is $s \to t \in \mathcal{S}_\sharp$ such that $\text{TCAP}^\mu_\mathcal{R}(t)$ and u' unify.*

We have the following.

Theorem 7 (Approximation of the CS Graph). *Let $\mathcal{R}$, $\mathcal{P}$ and $\mathcal{S}$ be TRSs and $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$. The* estimated *CS graph $\mathsf{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ contains the CS graph $\mathsf{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$.*

We also provide a computable definition of the SCC processor in Theorem 8.

Theorem 8 (SCC Processor using $\text{TCAP}^\mu_\mathcal{R}$). *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem. The CS processor Proc_{SCC} given by*

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

where

– *$\mathcal{S}_\mathcal{Q} = \varnothing$ if $\mathcal{Q}_\mathcal{X} = \varnothing$.*

– *$\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\rhd_\mu} \cup \{s \to t \mid s \to t \in \mathcal{S}_\sharp, \textsc{TCap}^\mu_{\mathcal{R}}(t)$ and u' unify for some $u' \to v' \in \mathcal{Q}\}$ if $\mathcal{Q}_{\mathcal{X}} \neq \varnothing$.*

is sound and complete.

Example 6. In Figure 1 (right) we show $\mathsf{EG}(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$ for $\mathcal{R}$ in Example 1. The graph has three SCCs $\mathcal{P}_1 = \{(1)\}$, $\mathcal{P}_2 = \{(8)\}$, and $\mathcal{P}_3 = \{(7),(14),(15)\}$. If we apply the CS processor Proc_{SCC} to the initial CS problem $(\mathsf{DP}(\mathcal{R},\mu),\mathcal{R},\mathsf{unh}(\mathcal{R},\mu),\mu^\sharp)$ for $(\mathcal{R},\mu)$ in Example 1, then we obtain the problems: $(\mathcal{P}_1,\mathcal{R},\varnothing,\mu^\sharp)$, $(\mathcal{P}_2,\mathcal{R},\varnothing,\mu^\sharp)$, $(\mathcal{P}_3,\mathcal{R},\mathcal{S}_3,\mu^\sharp)$, where $\mathcal{S}_3 = \{(16),(18),(19)\}$.

5.3 Reduction Triple Processor

A *μ-reduction pair* $(\gtrsim,\sqsupset)$ consists of a stable and μ-monotonic[3] quasi-ordering $\gtrsim$, and a well-founded stable relation $\sqsupset$ on terms in $\mathcal{T}(\mathcal{F},\mathcal{X})$ which are compatible, i.e., $\gtrsim \circ \sqsupset \subseteq \sqsupset$ or $\sqsupset \circ \gtrsim \subseteq \sqsupset$ [7].

In [7, 9], when a collapsing pair $u \to x$ occurs in a chain, we have to *look inside* the instantiated right-hand side $\sigma(x)$ for a μ-replacing subterm that, after marking it, does μ-rewrite to the (instantiated) left-hand side of another pair. For this reason, the quasi-orderings $\gtrsim$ of reduction pairs $(\gtrsim,\sqsupset)$ which are used in [7, 9] are required to have the μ-subterm property, i.e. $\unrhd_\mu \subseteq \gtrsim$. This is equivalent to impose $\mathsf{f}(x_1,\ldots,x_k) \gtrsim x_i$ for all projection rules $\mathsf{f}(x_1,\ldots,x_k) \to x_i$ with $\mathsf{f} \in \mathcal{F}$ and $i \in \mu(\mathsf{f})$. This is similar for markings: in [7] we have to ensure that $\mathsf{f}(x_1,\ldots,x_k) \gtrsim \mathsf{f}^\sharp(x_1,\ldots,x_k)$ for all defined symbols f in the signature. In [9], thanks to the notion of hidden term, we relaxed the last condition: we require $t \gtrsim t^\sharp$ for all (narrowable) *hidden terms t*. In [6], thanks to the notion of hiding context, we only require that $\gtrsim$ is compatible with the projections $\mathsf{f}(x_1,\ldots,x_k) \to x_i$ for those symbols f and positions i such that f *hides position i*. However, this information is implicitly encoded as (new) pairs $\mathsf{U}(\mathsf{f}(x_1,\ldots,x_k)) \to \mathsf{U}(x_i)$ in the set $\mathcal{P}$. The strict component $\sqsupset$ of the reduction pair $(\gtrsim,\sqsupset)$ is used with these new pairs now.

In this paper, since the rules in $\mathcal{S}$ are not considered as ordinary pairs (in the sense of [6, 9]) we can relax the conditions imposed to the orderings dealing with these rules. Furthermore, since rules in $\mathcal{S}$ are applied only once to the root of the terms, we only have to impose stability to the relation which is compatible with these rules (no transitivity, reflexivity, well-foundedness or μ-monotonicity is required).

Therefore, we can use *μ-reduction triples* $(\gtrsim,\sqsupset,\succeq)$ now, where $(\gtrsim,\sqsupset)$ is a μ-reduction pair and $\succeq$ is a stable relation on terms which is compatible with $\gtrsim$ or $\sqsupset$, i.e., $\succeq \circ \gtrsim \subseteq \gtrsim$ or $\sqsupset \circ \succeq \subseteq \sqsupset$.

Theorem 9 (μ-Reduction Triple Processor). *Let $\tau = (\mathcal{P},\mathcal{R},\mathcal{S},\mu)$ be a CS problem. Let $(\gtrsim,\sqsupset,\succeq)$ be a μ-reduction triple such that*

[3] A binary relation R on terms is μ-monotonic if for all terms $s,t,t_1,\ldots,t_k$, and k-ary symbols f, whenever $s\ \mathsf{R}\ t$ and $i \in \mu(\mathsf{f})$ we have $\mathsf{f}(t_1,\ldots,t_{i-1},s,\ldots,t_k)\ \mathsf{R}\ \mathsf{f}(t_1,\ldots,t_{i-1},t,\ldots,t_k)$.

1. $\mathcal{P} \subseteq \gtrsim \cup \sqsupset$, $\mathcal{R} \subseteq \gtrsim$, and
2. whenever $\mathcal{P}_{\mathcal{X}} \neq \varnothing$ we have that $\mathcal{S} \subseteq \gtrsim \cup \sqsupset \cup \succeq$.

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

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

is sound and complete.

Since rules from $\mathcal{S}$ are only applied after using a collapsing pair, we only need to make them compatible with some component of the triple if $\mathcal{P}$ contains collapsing pairs, i.e., if $\mathcal{P}_{\mathcal{X}} \neq \varnothing$. Another advantage is that we can now *remove rules from* $\mathcal{S}$. Furthermore, we can increase the power of this definition by considering the *usable rules* corresponding to $\mathcal{P}$, instead of $\mathcal{R}$ as a whole (see [6, 16]), and also by using argument filterings [9].

Example 7. (Continuing Example 6) Consider the CS problem $\tau = (\mathcal{P}_3, \mathcal{R}, \mathcal{S}_3, \mu^\sharp)$ where $\mathcal{P}_3 = \{(7), (14), (15)\}$, $\mathcal{S}_3 = \{(16), (18), (19)\}$ and $\mathcal{R}$ is the TRS in Example 1. If we apply Proc_{RT} to the CS problem τ by using the μ-reduction triple $(\geq, >, \geq)$ where $\geq$ and $>$ are the orderings induced by the following polynomial interpretation (see [17] for missing notation and definitions):

$[\mathsf{if}](x,y,z)$	$= (1/2 \times x) + y + z$	$[\mathsf{minus}](x,y)$	$= (2 \times x) + (2 \times y) + 1/2$
$[\mathsf{p}](x)$	$= (1/2 \times x)$	$[\mathsf{0}]$	$= 0$
$[\mathsf{false}]$	$= 0$	$[\mathsf{s}](x)$	$= (2 \times x) + 2$
$[\mathsf{true}]$	$= 2$	$[\mathsf{gt}](x,y)$	$= (2 \times x) + (1/2 \times y)$
$[\mathsf{M}](x,y)$	$= (2 \times x) + (2 \times y) + 1/2$	$[\mathsf{IF}](x,y,z)$	$= (1/2 \times x) + y + z$

then, we have $[\ell] \geq [r]$ for all (usable) rules in $\mathcal{R}$ and, for the rules in $\mathcal{P}_3$ and $\mathcal{S}_3$, we have

$$\begin{array}{rcl} [\mathsf{M}(x,y)] & \geq & [\mathsf{IF}(\mathsf{gt}(y,\mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)] \\ [\mathsf{IF}(\mathsf{true}, x, y)] & > & [x] \\ [\mathsf{IF}(\mathsf{false}, x, y)] & \geq & [y] \end{array} \qquad \begin{array}{rcl} [\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))] & \geq & [\mathsf{M}(\mathsf{p}(x), \mathsf{p}(y))] \\ [\mathsf{minus}(x,y)] & > & [y] \\ [\mathsf{minus}(x,y)] & > & [x] \end{array}$$

Then, we get $\mathsf{Proc}_{RT}(\tau) = \{(\{(7), (15)\}, \mathcal{R}, \{(16)\}, \mu^\sharp)\}$.

5.4 Subterm Processor

The subterm criterion was adapted to CSR in [7], but its use was restricted to noncollapsing pairs [7, Theorem 5]. In [9], a new version for collapsing pairs was defined, but in this version you can only remove all collapsing pairs and the projection π is restricted to μ-replacing positions. Our new version is fully general and able to remove collapsing and noncollapsing pairs at the same time. Furthermore, we are also able to remove rules in $\mathcal{S}$. Before introducing it, we need the following definition.

Definition 10 (Root Symbols of a TRS [9]). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS. The set of* root symbols *associated to $\mathcal{R}$ is:*

$$\mathsf{Root}(\mathcal{R}) = \{\mathsf{root}(\ell) \mid \ell \to r \in R\} \cup \{\mathsf{root}(r) \mid \ell \to r \in R, r \notin \mathcal{X}\}$$

Definition 11 (Simple Projection). *Let $\mathcal{R}$ be a TRS. A* simple projection *for $\mathcal{R}$ is a mapping π that assigns to every k-ary symbol $\mathsf{f} \in \mathsf{Root}(\mathcal{R})$ an argument position $i \in \{1, \ldots, k\}$. This mapping is extended to terms by*

$$\pi(t) = \begin{cases} t|_{\pi(\mathsf{f})} & \textit{if } t = \mathsf{f}(t_1, \ldots, t_k) \textit{ and } \mathsf{f} \in \mathsf{Root}(\mathcal{R}) \\ t & \textit{otherwise} \end{cases}$$

Theorem 10 (Subterm Processor). *Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem where $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$, $\mathcal{P} = (\mathcal{G}, P)$ and $\mathcal{S} = (\mathcal{H}, S)$. Assume that (1) $\mathsf{Root}(\mathcal{P}) \cap \mathcal{D} = \varnothing$, (2) the rules in $\mathcal{P}_{\mathcal{G}} \cup \mathcal{S}_{\sharp}$ are noncollapsing, (3) for all $s_i \to t_i \in \mathcal{S}_{\rhd_\mu}$, $\mathsf{root}(s_i), \mathsf{root}(t_i) \notin \mathsf{Root}(\mathcal{P})$ and (4) for all $s_i \to t_i \in \mathcal{S}_{\sharp}$, $\mathsf{root}(s_i) \notin \mathsf{Root}(\mathcal{P})$ and $\mathsf{root}(t_i) \in \mathsf{Root}(\mathcal{P})$. Let π be a simple projection for $\mathcal{P}$. Let $\mathcal{P}_{\pi,\rhd_\mu} = \{u \to v \in \mathcal{P} \mid \pi(u) \rhd_\mu \pi(v)\}$ and $\mathcal{S}_{\pi,\rhd_\mu} = \{s \to t \in \mathcal{S} \mid \pi(s) \rhd_\mu \pi(t)\}$. Then, $\mathsf{Proc}_{subterm}$ given by*

$$\mathsf{Proc}_{subterm}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi,\rhd_\mu}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi,\rhd_\mu}, \mu)\} & \textit{if } \pi(\mathcal{P}) \subseteq \unrhd_\mu \\ & \textit{and whenever } \mathcal{P}_{\mathcal{X}} \neq \varnothing, \\ & \textit{then } \pi(\mathcal{S}) \subseteq \unrhd_\mu \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \textit{otherwise} \end{cases}$$

is sound and complete.

Notice that the conditions in Theorem 10 are not harmful in practice because the CS problems which are obtained from CS-TRSs normally satisfy those conditions.

Example 8. (Continuing Example 7) We have the CS problem $(\mathcal{P}_5, \mathcal{R}, \mathcal{S}_5, \mu^\sharp)$ where $\mathcal{P}_5 = \{(7), (15)\}$ and $\mathcal{S}_5 = \{(16)\}$. We can apply the subterm processor $\mathsf{Proc}_{subterm}$ by using the projection $\pi(\mathsf{IF}) = 3$ and $\pi(\mathsf{M}) = 1$:

$$\begin{aligned} \pi(\mathsf{M}(x,y)) = x &\unrhd_\mu x = \pi(\mathsf{IF}(\mathsf{gt}(y,\mathsf{0}), \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)), x)) \\ \pi(\mathsf{IF}(\mathsf{false}, x, y)) = y &\unrhd_\mu y = \pi(y) \\ \pi(\mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y))) = \mathsf{minus}(\mathsf{p}(x), \mathsf{p}(y)) &\rhd_\mu \mathsf{p}(x) = \pi(\mathsf{M}(\mathsf{p}(x), \mathsf{p}(y))) \end{aligned}$$

We obtain the CS problem $\tau' = (\{(7), (15)\}, \mathcal{R}, \varnothing, \mu)$ for which we can use Proc_{SCC} to conclude that there is no cycle, i.e., $\mathsf{Proc}_{SCC}(\tau') = \varnothing$.

6 Using the CSDP Framework in Maude

Proving termination of programs in sophisticated equational languages like OBJ, CafeOBJ or Maude is difficult because these programs combine different features that are not supported by state-of-the-art termination tools. For instance, the following Maude program combines the use of an evaluation strategy and types given as sorts in the specification [3].

```
fmod LengthOfFiniteLists is
   sorts Nat NatList NatIList .
   subsort NatList < NatIList .
   op 0 : -> Nat .
   op s : Nat -> Nat .
   op zeros : -> NatIList .
   op nil : -> NatList .
   op cons : Nat NatIList -> NatIList [strat (1 0)] .
   op cons : Nat NatList -> NatList [strat (1 0)] .
   op length : NatList -> Nat .
   vars M N : Nat .
   var IL : NatIList .
   var L : NatList .
   eq zeros = cons(0,zeros) .
   eq length(nil) = 0 .
   eq length(cons(N, L)) = s(length(L)) .
endfm
```

Nowadays, MU-TERM [14, 13] can separately prove termination of order-sorted rewriting [18] and CSR, but it is not able to handle programs which combine both of them. Then, we use the transformation developed in [3] to transform this system into a CS-TRS (without sorts). Such a CS-TRS can be found in the Termination Problems Data Base[4] (TPDB): `TRS/CSR_Maude/LengthOfFinite-Lists_complete.trs`. As far as we know, MU-TERM is the only tool that can prove termination of this system thanks to the CSDP framework presented in this paper[5].

7 Experimental Evaluation

From Friday to Saturday, December 18-19, 2009, the 2009 International Termination Competition took place and a CSR termination category was included. In the termination competition, the benchmarks are executed in a completely automatic way with a timeout of 60 seconds over a subset of 37 systems[6] of the complete collection of the 109 CS-TRSs of the TPDB 7.0.

The results in this paper have been implemented as part of the termination tool MU-TERM. Our tool MU-TERM participated in the aforementioned CSR category of the 2009 Termination Competition. The results of the competition are summarized in Table 1. Tools AProVE [19] and VMTL [20] implement the context-sensitive dependency pairs using the transformational approach in [6]. The techniques implemented by Jambox [21] to prove termination of CSR are not documented yet, to our knowledge. As showed in Table 1, we are able to prove the same number of systems than AProVE, but MU-TERM is almost two

[4] `http://www.lri.fr/~marche/tpdb/`

[5] On May 12, 2010, we introduced this system in the online version of AProVE `http://aprove.informatik.rwth-aachen.de/`, and a timeout occurred after 120 seconds (maximum timeout). MU-TERM proof can be found in `http://zenon.dsic.upv.es/muterm/benchmarks/benchmarks-csr/benchmarks.html`

[6] See `http://termcomp.uibk.ac.at/termcomp/competition/competitionResults.seam?category=10230\&competitionId=101722\&actionMethod=competition\%2FcategoryList.xhtml\%3AcompetitionCategories.forward\&conversationPropagation=begin`

Table 1. 2009 Termination Competition Results (Context-Sensitive Rewriting)

Tool Version	Proved	Average time
AProVE	34/37	3.084s
Jambox	28/37	2.292s
MU-TERM	34/37	1.277s
VMTL	29/37	6.708s

and a half times faster. Furthermore, we prove termination of 95 of the 109 examples. To our knowledge, there is no tool that can prove more than those 95 examples from this collection of problems. And, as remarked in Section 6, there are interesting examples which can be handled by MU-TERM only.

We have also executed the complete collection of systems of the CSR category[7], where we compare the 2009 and 2007 competition versions of MU-TERM. In the 2007 version, the CSDP framework was not available. Now, we can prove 15 more examples and, when comparing the execution times which they took over the 80 examples where both tools succeeded ($84,48$ seconds vs. $15,073$ seconds), we are more than $5,5$ times faster now.

8 Related Work

In [6], a transformation of collapsing pairs into 'ordinary' (i.e., noncollapsing) pairs is introduced by using the new notion of *hiding context* [6, Definition 7]. We easily and naturally included such a transformation as a new processor Proc_{eColl} in our framework (see Theorem 4). The claimed advantage of [6] is that the notion of chain is simplified to Item 1 in Definition 5. But, although the definition of chain in [6] is apparently closer to the standard one [12, Definition 3], this does *not* mean that we can use or easily 'translate' existing DP-processors (see [12]) to be used with CSR. Besides the narrowing processor in [9, Theorem 16], the reduction pair processor with usable rules in [6, Theorem 21] is a clear example, because the avoidance of collapsing pairs does not improve the previous results about usable rules for CSR investigated in [16].

As we have seen in this paper, collapsing pairs are an essential part of the theoretical description of termination of CSR. Actually, the transformational approach in [6] *explicitly* uses them for introducing the new unhiding pairs in [6, Definition 9]. This shows that the most basic notion when *modeling* the termination behavior of CSR is that of collapsing pair and that unhiding pairs should be better considered as an ingredient for handling collapsing pairs in proofs of termination (as implemented by processor Proc_{eColl} above). Furthermore, the application of such a transformation in the very beginning of the termination analysis of CS-TRSs (as done in [6]) typically leads to obtain a more complex dependency graph (see in Figure 1 (left)) which, as witnessed by our experimental

[7] A complete report of our experiments can be found in `http://zenon.dsic.upv.es/muterm/benchmarks/`

analysis in Section 7, can be more difficult to analyze when proving termination in practice.

Our approach clarifies the role of collapsing pairs to model the termination behavior of CSR. Furthermore, the new notions introduced in this paper lead to a more 'robust' framework. For instance, in order to integrate in [6] the new improvement in the notion of hiding context (see Definition 2), one has to *redefine* the notion of context-sensitive dependency pair in [6]. In our approach, the context-sensitive dependency pairs are always the same.

9 Conclusions

When proofs of termination of CSR are mechanized following the context-sensitive dependency pair approach [7], handling collapsing pairs is difficult. In [6] this problem is solved by a transformation which disregards collapsing pairs (so we loose their descriptive power), adds a new fresh symbol U which has nothing to do with the original CS-TRS, and makes the dependency graph harder to understand.

We have shown a different way to mechanize the context-sensitive dependency pair approach. The idea is adding a new TRS, the *unhiding TRS*, which avoids the extra requirements in [7]. Thanks to the flexibility of our framework, we can use all existing processors in the literature, improve the existing ones by taking advantage of having collapsing pairs, and define new processors. Furthermore, we have improved the notion of *hide* given in [6]. Our experimental evaluation shows that our techniques lead to an implementation which offers the best performance in terms of solved problems and efficiency.

References

1. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. Journal of Functional and Logic Programming **1998**(1) (1998) 1–61
2. Bruni, R., Meseguer, J.: Semantic Foundations for Generalized Rewrite Theories. Theoretical Computer Science **360**(1) (2006) 386–414
3. Durán, F., Lucas, S., Marché, C., Meseguer, J., Urbain, X.: Proving Operational Termination of Membership Equational Programs. Higher-Order and Symbolic Computation **21**(1-2) (2008) 59–88
4. Endrullis, J., Hendriks, D.: From Outermost to Context-Sensitive Rewriting. In Treinen, R., ed.: Proc. of 20th International Conference on Rewriting Techniques and Applications, RTA'09. Volume 5595 of Lecture Notes in Computer Science., Springer-Verlag (2009) 305–319
5. Fernández, M.L.: Relaxing Monotonicity for Innermostt Termination. Information Processing Letters **93**(3) (2005) 117–123
6. Alarcón, B., Emmes, F., Fuhs, C., Giesl, J., Gutiérrez, R., Lucas, S., Schneider-Kamp, P., Thiemann, R.: Improving Context-Sensitive Dependency Pairs. In Cervesato, I., Veith, H., Voronkov, A., eds.: Proc. of 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08. Volume 5330 of Lecture Notes in Computer Science., Springer-Verlag (2008) 636–651

7. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. In Arun-Kumar, S., Garg, N., eds.: Proc. of 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06. Volume 4337 of Lecture Notes in Computer Science., Springer-Verlag (2006) 297–308
8. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science **236**(1–2) (2000) 133–178
9. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. Information and Computation **To appear** (2010)
10. Gutiérrez, R., Lucas, S.: Proving Termination in the Context-Sensitive Dependency Pairs Framework. Technical report, Universidad Politécnica de Valencia (February 2010) Available as Technical Report DSIC-II/02/10.
11. Ohlebusch, E.: Advanced Topics in Term Rewriting. Springer-Verlag (2002)
12. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. Journal of Automatic Reasoning **37**(3) (2006) 155–203
13. Lucas, S.: MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In Oostrom, V.v., ed.: Proc. of 15th International Conference on Rewriting Techniques and Applications, RTA'04. Volume 3091 of Lecture Notes in Computer Science., Springer-Verlag (2004) 200–209 Available at `http://zenon.dsic.upv.es/muterm/`.
14. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. Electronic Notes in Theoretical Computer Science **188** (2007) 105–115
15. Hirokawa, N., Middeldorp, A.: Automating the Dependency Pair Method. Information and Computation **199** (2005) 172–199
16. Gutiérrez, R., Lucas, S., Urbain, X.: Usable Rules for Context-Sensitive Rewrite Systems. In Voronkov, A., ed.: Proc. of 19th International Conference on Rewriting Techniques and Applications, RTA'08. Volume 5117 of Lecture Notes in Computer Science., Springer-Verlag (2008) 126–141
17. Lucas, S.: Polynomials over the Reals in Proofs of Termination: from Theory to Practice. RAIRO Theoretical Informatics and Applications **39**(3) (2005) 547–586
18. Lucas, S., Meseguer, J.: Order-Sorted Dependency Pairs. In Antoy, S., Albert, E., eds.: Proc. of 10th International ACM SIGPLAN Sympsium on Principles and Practice of Declarative Programming, PPDP'08, ACM Press (2008) 108–119
19. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In Furbach, U., Shankar, N., eds.: Proc. of 3rd International Joint Conference on Automated Reasoning, IJCAR'06. Volume 4130 of Lecture Notes in Artificial Intelligence., Springer-Verlag (2006) 281–286 Available at `http://www-i2.informatik.rwth-aachen.de/AProVE`.
20. Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In Treinen, R., ed.: Proc. of 20th International Conference on Rewriting Techniques and Applications, RTA'09. Volume 5595 of Lecture Notes in Computer Science., Springer-Verlag (2009) 285–294
21. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewriting (2009) Available at `http://joerg.endrullis.de/jambox.html`.