

UNIVERSIDAD POLITÉCNICA DE VALENCIA  
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN  
DOCTORADO EN INFORMÁTICA

PH.D. THESIS

# Automatic Proofs of Termination of Context-Sensitive Rewriting

CANDIDATE:  
Raúl Gutiérrez

SUPERVISOR:  
Salvador Lucas

September 2010

Work partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and TIN 2007-68118-C02.

---

Author's address:

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

to my father, my mother,  
my grandmother and my grandfather.



---

# Acknowledgments

I would like to first thank María for giving me the opportunity to develop my PhD thesis in Valencia. Big thanks to Salva, the architect of my academical knowledge and my thesis. I can only feel gratitude to him for guiding me always through the correct way.

I also thank to all the members of the ELP/MIST group, and particularly to my lab mates during all these years: Bea, Toni, Rafa, Pepe, Gustavo, Dani, Alexei, Mauricio, Tama, Marco, Sonia, Fernando. . . All of them made me feel like in a family.

Many thanks to my coauthors: especially to Bea, for supporting me during all these years and being always there; to Pepe, for showing me the passion of doing things in the best way; and to Rafa, that always made me laugh.

Especial thanks to my lunch mates: to Germán, because there is always something to learn from him; to Pepe, because he wrote many lines of Haskell code in my mind; to Marco, for his happiness and passion; to Josep, a good neighbour and a good friend; to Cèsar, a model to follow giving lectures; to Santi, for sharing always his experience; and many others during the last years.

Also thank to all the people I have visited and that allowed me to learn from them. To Xavier, because it is always a pleasure to work with him. To all the group in the RWTH Aachen, my second academical home. Thanks to Jürgen for giving me the opportunity to work with the AProVE team: thanks to René, to Peter, to Stephan, and Carsten. Also thanks to the rest of people in the department for all these lunches and coffee hours: to Jonathan, Stefan, Carsten, Daniel, Martin, Tingting, Viet Yen, . . .

To all the authors of all the papers and books that I have read during my PhD, that allowed me to enrich my knowledge, and that showed me the pleasure of research.

Finally, and the most important, thanks to Alicia, any character written in this thesis is thanks to you, to your patience and your love. Thank you.

September 2010,

Raúl Gutiérrez



---

# Abstract

The idea of an incremental application of different termination techniques as *processors* for solving termination problems has shown to be a powerful and efficient way to prove termination of rewriting. Nowadays, the *dependency pair framework* (which develops this idea) is the most successful approach for proving termination of rewriting. The dependency pair framework relies on the notion of dependency pair to decompose a termination problem into a set of *dependency pair problems*. These dependency pair problems can be treated independently by applying different *dependency pair processors*. If we prove (disprove) the *finiteness* of all (some) of the dependency pair problems, then we can ensure that the system is terminating (nonterminating). This simple but powerful schema is the basis of state-of-art tools for automatically proving termination of rewriting.

*Context-sensitive rewriting* [Luc98, Luc02] is a restriction of rewriting that forbids reductions on some subexpressions and that has proven to be useful in modeling and analyzing programming language features at different levels. In particular, termination of context-sensitive rewriting has proven to be useful in analyzing and proving termination of programs in several programming languages and variants of term rewriting. Over the last fifteen years, a number of techniques for proving termination of context-sensitive rewriting have been developed and implemented (essentially transformations and appropriate orderings). However, the definition of a *context-sensitive dependency pair framework* started only four years ago, when the research developed in this thesis began.

In this thesis, we show how to develop a dependency pair framework for proving termination of context-sensitive rewriting. We also provide experimental evidence of the advantages of the context-sensitive dependency pair framework in the development of tools for automatically proving termination of context-sensitive rewriting.



---

# Resumen

La idea de aplicar de forma incremental diferentes técnicas de terminación encapsuladas como *procesadores* con el objetivo de resolver problemas de terminación se está mostrando como una técnica eficiente y potente de probar la terminación de la reescritura. Hoy en día, el *marco de pares de dependencia* (que desarrolla esta idea) es la aproximación más exitosa para probar la terminación de la reescritura. El marco de pares de dependencia utiliza la noción de par de dependencia para descomponer un problema de terminación en un conjunto de *problemas de pares de dependencia*. Estos problemas de pares de dependencia pueden ser tratados de manera independiente aplicando diferentes *procesadores de pares de dependencia*. Si conseguimos probar la *finitud* de todos los problemas de terminación, entonces podemos asegurar que el sistema es terminante. Si conseguimos refutar la finitud de alguno de los problemas de terminación, entonces podemos asegurar que el sistema es no-terminante. Este sencillo y a la par potente esquema es la base del estado del arte de las herramientas que prueban de forma automática la terminación de la reescritura.

La *reescritura sensible al contexto* [Luc98, Luc02] es una restricción de la reescritura que prohíbe las reducciones de algunas subexpresiones y que se ha demostrado que es útil para modelar y analizar propiedades de los lenguajes de programación a distintos niveles. En particular, la terminación de la reescritura sensible al contexto es útil para analizar y probar la terminación de programas en varios lenguajes de programación y variantes de sistemas de reescritura de terminos. En los últimos quince años se han desarrollado y programado muchas técnicas para probar la terminación de la reescritura sensible al contexto (fundamentalmente transformaciones y órdenes). Sin embargo, la definición de *par de dependencia sensible al contexto* y de su marco comenzó sólo hace unos cuatro años, cuando se inició el desarrollo de esta tesis.

En esta tesis mostramos como desarrollar un marco de pares de dependencia para probar la terminación de la reescritura sensible al contexto y mostramos resultados experimentales de las ventajas del marco de pares de dependencia sensible al contexto en el desarrollo de herramientas para probar automáticamente la terminación de la reescritura sensible al contexto.



---

# Resum

La idea d'aplicar de forma incremental diferents tècniques de terminació encapsulades com *processadors* amb l'objectiu de resoldre problemes de terminació s'està mostrant com una tècnica eficient i potent de provar la terminació de la reescriptura. Avui dia, el *marc de parells de dependència* (que desenvolupa aquesta idea) és l'aproximació més reeixida per a provar la terminació de la reescriptura. El marc de parells de dependència utilitza la noció de parell de dependència per a descompondre un problema de terminació en un conjunt de *problemes de parells de dependència*. Aquests problemes de parells de dependència poden ser tractats de manera independent aplicant diferents processadors de parells de dependència. Si aconseguim provar la *finitud* de tots els problemes de terminació, llavors podem assegurar que el sistema és terminant. Si aconseguim refutar la finitud d'algun dels problemes de terminació, llavors podem assegurar que el sistema és no-terminant. Aquest senzill i potent esquema és la base de l'estat de l'art de les eines que proven de forma automàtica la terminació de la reescriptura.

La *reescriptura sensible al context* [Luc98, Luc02] és una restricció de la reescriptura que prohibeix les reduccions d'algunes subexpressions i que s'ha demostrat que és útil per a modelar i analitzar propietats dels llenguatges de programació a diferents nivells. En particular, la terminació de la reescriptura sensible al context és útil per a analitzar i provar la terminació de programes en diversos llenguatges de programació i variants de sistemes de reescriptura de termes. En els últims quinze anys s'han desenvolupat i programat moltes tècniques per a provar la terminació de la reescriptura sensible al context (fonamentalment transformacions i ordres). No obstant això, la definició de *parell de dependència sensible al context* i del seu marc va començar només fa uns quatre anys, quan es va iniciar el desenvolupament d'aquesta tesi.

En aquesta tesi anem a mostrar com desenvolupar un marc de parells de dependència per a provar la terminació de la reescriptura sensible al context i anem a mostrar resultats experimentals dels avantatges del marc de parells de dependència sensible al context en el desenvolupament d'eines per a provar automàticament la terminació de la reescriptura sensible al context.



---

# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Termination and Term Rewriting . . . . .	2
1.1.1	Dependency Pairs . . . . .	4
1.1.2	Dependency Pair Framework . . . . .	6
1.2	Context-Sensitive Rewriting . . . . .	7
1.2.1	Applications of Termination of Context-Sensitive Rewriting . . . . .	11
1.2.2	Techniques for Proving Termination of CSR Automatically . . . . .	14
1.3	Contributions of the Thesis . . . . .	14
1.4	Plan of the Thesis . . . . .	15
<b>I</b>	<b>Summary of the Research</b>	<b>19</b>
<b>2</b>	<b>Preliminaries</b>	<b>21</b>
2.1	Abstract Reduction Systems . . . . .	21
2.2	Signatures, Terms, and Positions . . . . .	21
2.3	Substitutions, Renamings, and Unifiers . . . . .	22
2.4	Binary Relations over Terms . . . . .	23
2.5	Rewrite Systems and Term Rewriting . . . . .	23
2.6	Narrowing . . . . .	24
2.7	Context-Sensitive Rewriting . . . . .	24
<b>3</b>	<b>Structure of Infinite <math>\mu</math>-Rewrite Sequences</b>	<b>27</b>
3.1	Minimal Non- $\mu$ -Terminating Terms . . . . .	28
3.2	Infinite $\mu$ -Rewrite Sequences Starting from Minimal Terms . . . . .	29
3.3	Hidden Terms . . . . .	31
3.4	Infinite $\mu$ -Rewrite Sequences Starting from Strongly Minimal Terms . . . . .	35
3.5	Historical Development of Infinite $\mu$ -Rewrite Sequences . . . . .	36

---

<b>4</b>	<b>Context-Sensitive Dependency Pairs</b>	<b>39</b>
4.1	Defining CSDPs . . . . .	40
4.2	Unhiding TRS . . . . .	42
4.3	Chains of CSDPs . . . . .	45
4.4	Historical Development of CSDPs . . . . .	46
<b>5</b>	<b>Context-Sensitive Dependency Pair Framework</b>	<b>51</b>
5.1	Historical Development of the CSDP Framework . . . . .	53
<b>6</b>	<b>CS Processors</b>	<b>55</b>
6.1	Preprocessing for Rule Removal . . . . .	55
6.2	Context-Sensitive (Dependency) Graph . . . . .	57
6.2.1	Definition of the Context-Sensitive Graph . . . . .	58
6.2.2	Estimating the Context-Sensitive Graph . . . . .	58
6.2.3	Historical Development of the CSDG . . . . .	61
6.3	Basic Processors . . . . .	64
6.4	Treating Collapsing Pairs . . . . .	67
6.5	Historical Development of the Collapsing Pair Transformation . . . . .	68
6.6	Reduction Triple Processor . . . . .	68
6.6.1	Historical Development of $\mu$ -Reduction Triples . . . . .	70
6.7	Reduction Triple Processor with Usable Rules . . . . .	71
6.7.1	Historical Development of $\mu$ -Reduction Triples with Usable Rules . . . . .	78
6.8	Subterm Criterion . . . . .	80
6.8.1	Historical Development of Subterm Criterion for CSR . . . . .	84
<b>7</b>	<b>Implementing the CSDP Framework</b>	<b>87</b>
7.1	The Language . . . . .	87
7.2	Term Rewriting Systems . . . . .	88
7.3	CSDP Framework . . . . .	90
7.3.1	Problems . . . . .	90
7.3.2	Proof . . . . .	91
7.3.3	Processors . . . . .	92
7.3.4	Strategy . . . . .	93
<b>8</b>	<b>Experiments</b>	<b>95</b>
8.1	2007 Termination Competition . . . . .	95
8.2	2009 Termination Competition . . . . .	96
8.3	CS Processor Evaluation . . . . .	97

---

8.4 Contributions of the CSDP Framework to MU-TERM . . . . .	98
8.5 Advantages of the CSDP Framework over Previous Approaches	99
<b>Conclusions</b>	<b>101</b>
<b>Bibliography</b>	<b>113</b>
<b>Index</b>	<b>127</b>
<b>II Publications Associated to the Thesis</b>	<b>129</b>
<b>List of Publications</b>	<b>131</b>
<b>Publications (full text)</b>	<b>133</b>
8.6 Context-Sensitive Dependency Pairs . . . . .	133
8.7 Improving the Context-Sensitive Dependency Graph . . . . .	146
8.8 Proving Termination of Context-Sensitive Rewriting with MU- TERM . . . . .	160
8.9 Usable Rules for Context-Sensitive Rewrite Systems . . . . .	172
8.10 Improving Context-Sensitive Dependency Pairs . . . . .	189
8.11 Context-Sensitive Dependency Pairs . . . . .	206
8.12 Proving Termination in the CSDP Framework . . . . .	254
8.13 Proving Termination Properties with MU-TERM . . . . .	272
<b>Extended Versions</b>	<b>281</b>
8.14 Proving Termination in the CSDP Framework . . . . .	281



---

## List of Figures

1.1	Computing Wallis' approximation to $\frac{\pi}{2}$ . . . . .	10
3.1	Delayed function call introduced by the rule $a \rightarrow c(f(a))$ . . . . .	31
3.2	The delayed function call is activated by the application of the rule $f(c(x)) \rightarrow x$ . . . . .	33
3.3	The delayed function call $b$ is introduced by the application of the rule $a \rightarrow f(c(b))$ . . . . .	34
3.4	The context of the delayed function call $b$ is modified by the application of the rule $f(x) \rightarrow h(d(x))$ . . . . .	35
4.1	CSDPs of the CS-TRS in Example 10 . . . . .	41
6.1	Estimated CS Dependency Graph for $\mathcal{R}$ in Example 10 . . . . .	60
6.2	Estimated CS Dependency Graph for $\mathcal{R}$ in Example 9 . . . . .	61
6.3	Estimated CS Graph of Pairs from Example 52 following [AGL06] . . . . .	63
6.4	Estimated CS Graph of Pairs from Example 52 following [AEF+08] . . . . .	64
6.5	Estimated CS Dependency Graph from $\tau_0$ in Example 89 . . . . .	77



---

## List of Tables

8.1	2007 International Termination Competition results	96
8.2	2009 International Termination Competition results	97
8.3	Summary of processors used in MU-TERM	97
8.4	Comparison among MU-TERM versions	99



---

# 1

## Introduction

*“She lives beyond the grace of God, a wanderer in the outer darkness. She is ‘vampyr’, ‘nosferatu’. These creatures do not die like the bee after the first sting, but instead grow strong and become immortal once infected by another nosferatu. So, my friends we fight not one beast but legions that go on age after age after age, feeding on the blood of the living.”*

**Van Helsing**

*Bram Stoker’s Dracula, 1992.*

Immortality, a blessing or a condemnation. In computer science, we can think of immortality as *nontermination*. When a user executes a computer program she expects a response in finite time. The fact of getting no answer is usually due to an infinite computation. Such ‘immortal’ computations will consume the resources of the computer eternally, like a vampire, directly affecting the performance and the behavior of the system. Preventing this problem is crucial to obtain robust software.

*Termination* is a computational property that allows us to ensure that every computation in a program is always finite. In computability theory, it is associated to the so-called *halting problem*. Actually, it is not uncommon for a computer to be blocked by an abnormal and undesired infinite computation in the running software. Ensuring that any program at any moment and with any possible input will reach a final state (i.e., ensuring its termination) is one of the most important tasks for a software developer.

As proved by Alan Turing in 1936 for Turing machines, termination of programs is *undecidable*. This means that there is no general algorithm to decide whether a program is terminating or not. The menace of ‘vampire programs’ then is out there hidden in the shadows, waiting for the moment to infect our computer. However, the undecidability of termination does not mean that we

cannot develop algorithms and techniques (apart from garlic, crucifixes and wooden stakes, of course) that can succeed in proving termination of a specific particular program. However, those algorithms only succeed on restricted subsets of programs. A number of successful results and techniques for proving termination come from the area of *Term Rewriting* [BN98, Ohl02, TeR03], being the base, in many cases, of notions and techniques which are useful for proving termination of programs written in other programming languages.

The origins of term rewriting go back to the origins of computer science itself [Thu10, ST00]. Rewriting techniques were used in mathematics and mathematical logic as a suitable framework to analyze computational properties as confluence and termination of  $\lambda$ -calculus and combinatory logic in the third and fourth decade of the 20th century [Sch24, Chu32, CR36, New42]. Term rewriting is an operational principle that can be used in software engineering to implement and analyze abstract specifications (e.g., equationally-specified abstract data types [Gut75]), as the foundational principle in the development of programming languages [McC63, Han94, BJM00], in automatically proving termination of programs [GSSKT06, SKGAR07, AAC<sup>+</sup>08], in equational unification [Hul80], etc. In other words, term rewriting has applications in practical computer science, theoretical computer science, and mathematics.

## 1.1 Termination and Term Rewriting

*Term rewriting* is a branch of theoretical computer science that is based on equational logic. Equations are mathematical statements that allow us to state that two things are equivalent. For instance, the equation  $2 + 3 = 5$  states that 5 is equivalent to the expression  $2 + 3$ . In term rewriting, the equations are oriented from left to right. Oriented equations are called *rules* and written  $\ell \rightarrow r$  rather than  $\ell = r$ . The replacement, within a term  $t$ , of an instance  $\sigma(\ell)$  of the left-hand side  $\ell$  by the instance  $\sigma(r)$  of the right-hand side  $r$  is called (one-step) term rewriting for a given substitution  $\sigma$ . And the final outcome of the rewriting process (i.e., when we cannot apply more rules) is called a *normal form* and is considered as the “result” of the Turing complete computational mechanism which we call *term rewriting*.

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

also occur in  $\ell$ .

The research on techniques for *automatic termination analysis* of programs is really important for any kind of programming language: from functional or logic programming languages [GSSKT06, SKGAR07] to imperative languages [AAC<sup>+</sup>08]. The development of tools to *automatically* check the termination of programs is crucial in modern software development. In many cases, term rewriting gives us a suitable theoretical and practical framework for reasoning about termination.

Termination is also paramount in term rewriting systems. For a finite terminating rewrite system, a normal form of a given term can be found by a simple depth-first search. If the system is also confluent<sup>1</sup>, the normal form is unique. Termination of term rewriting is undecidable, as proved by Huet and Lankford [HL78].

Throughout the history of termination, different methods for proving termination of rewriting have been developed. The use of well-founded orderings is fundamental. This was anticipated by Iturriaga [Itu67] and Manna and Ness [MN70].

**Theorem 1 (Manna and Ness [MN70])** *A TRS  $(\mathcal{F}, R)$  is terminating if and only if there is a strict and well-founded ordering  $>$  on terms such that, for all terms  $s, t$ ,  $s \rightarrow t$  implies  $s > t$ .*

The problem with this theorem was that it is not well-suited for automation because, in general, an infinite number of comparisons between terms must be performed. This idea was refined by Lankford considering as many comparisons as there are rules in the TRS. As a counterpart, we have to use *reduction orderings*, i.e., well-founded, monotonic, and stable (under substitution) orderings on terms.

**Theorem 2 (Lankford [Lan79])** *A TRS  $(\mathcal{F}, R)$  is terminating if and only if there is a reduction ordering  $>$  over terms such that  $\ell > r$  for all  $\ell \rightarrow r \in R$ .*

The so-called Lankford theorem states that in order to prove a TRS  $\mathcal{R}$  terminating, we have to find a reduction ordering which is *compatible* with the rules in  $\mathcal{R}$  (i.e.,  $\ell > r$  for each rule  $\ell \rightarrow r \in \mathcal{R}$ ). This approach dominated the development of automatic techniques for proving termination of rewriting until 1995, and automatically generated reduction orderings like recursive path orderings [Der79, KNS85], polynomial orderings [Lan79, CL87], Knuth-Bendix orderings [KB70], and others [Les82] were used in practice. However, these

---

<sup>1</sup>A TRS is confluent if for every term  $s, t$  and  $t'$ , whenever  $s$  rewrites to both  $t$  and  $t'$ , there is a term  $u$  such that both  $t$  and  $t'$  rewrite to  $u$ .

orderings correspond to the class of *simplification orderings*<sup>2</sup>, which is quite a restricted class of orderings.

---

### Example 3

Consider the following one-rule TRS  $\mathcal{R}$  [BN98]:

$$f(f(x)) \rightarrow f(g(f(x)))$$

We know that  $\mathcal{R}$  is terminating, but it is not simply terminating (see [BN98, Example 5.4.9]) and, hence, we cannot find a simplification ordering that can prove this system terminating.

---

Furthermore, termination under such simplification orderings is also undecidable [MG95]. In order to improve this situation, the idea of considering the function calls instead of the rules for proving termination was growing in the community. In 1996, Thomas Arts introduced the notion of *dependency pair*, a notion that changed the way of thinking about termination [Art96]. The *dependency pair approach* [AG00], which is based on capturing the recursive calls of a system to decompose the proof of termination into simpler constraints became the most widely used technique for automatically proving termination of Term Rewriting Systems.

#### 1.1.1 Dependency Pairs

The dependency pair (DP) technique focuses on the following idea: the rules that are really able to produce infinite sequences are those rules  $\ell \rightarrow r$  such that  $r$  contains some *defined* symbol<sup>3</sup>  $g$ . Intuitively, we can think of these rules as representing some possible (direct or indirect) recursive calls. Such recursion paths associated to each rule  $\ell \rightarrow r$  are represented as new rules  $u \rightarrow v$ , where  $u = F(l_1, \dots, l_k)$  if  $\ell = f(l_1, \dots, l_k)$ , and where  $v = G(s_1, \dots, s_m)$  if  $s = g(s_1, \dots, s_m)$  is a subterm of  $r$  and  $g$  is a defined symbol.  $F$  and  $G$  are fresh symbols associated to defined symbols  $f$  and  $g$ , respectively, which are intended to identify the possible calls. For this reason, the DP technique starts by considering a new TRS  $\text{DP}(\mathcal{R})$  that contains all these new rules  $u \rightarrow v$  for each  $\ell \rightarrow r \in \mathcal{R}$ .

---

<sup>2</sup>A simplification ordering is a stable and monotonic ordering  $>$  which satisfies the subterm property, that is: for all terms  $s$ , and strict subterms  $t$  of  $s$ , we have  $s > t$  [Der79].

<sup>3</sup>A symbol  $g \in \mathcal{F}$  is defined in  $\mathcal{R}$  if there is a rule in  $\mathcal{R}$  whose left-hand side is of the form  $g(l_1, \dots, l_k)$ .

**Example 4**

Continuing with Example 3, we have the following set of DPs  $\text{DP}(\mathcal{R})$ <sup>4</sup>:

$$\begin{aligned} F(f(x)) &\rightarrow F(g(f(x))) \\ F(f(x)) &\rightarrow F(x) \end{aligned}$$

The rules in  $\mathcal{R}$  and the rules in  $\text{DP}(\mathcal{R})$  determine the so-called *dependency chains* whose finiteness characterizes termination of  $\mathcal{R}$  [AG00]. A *chain of DPs* is a sequence  $u_i \rightarrow v_i$  of DPs together with a substitution  $\sigma$  such that  $\sigma(v_i)$  rewrites to  $\sigma(u_{i+1})$  for all  $i \geq 1$  (as usual, we assume variable renaming if  $v_i$  and  $u_{i+1}$  are not variable disjoint).

**Theorem 5** [AG00] *A TRS  $\mathcal{R}$  is terminating iff there exists a weakly monotonic quasi-ordering  $\geq$ , where both  $\geq$  and its strict part  $>$  are closed under substitutions, such that  $>$  is well-founded and*

- $\ell \geq r$  for all rules  $\ell \rightarrow r \in \mathcal{R}$  and
- $u > v$  for all DPs  $u \rightarrow v$ .

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

**Example 6**

The TRS  $\mathcal{R}$  from Example 3 together with its DPs (shown in Example 4) is proved terminating by using the following polynomial interpretation, which interprets each function symbol as a polynomial over the naturals:

$$\begin{aligned} [F](x) &= x & [f](x) &= x + 1 \\ [g](x) &= 0 \end{aligned}$$

Then, we have the following:

$$\begin{aligned} [f(f(x))] &= x + 2 \geq 1 = [f(g(f(x)))] \\ [F(f(x))] &= x + 1 > 0 = [F(g(f(x)))] \\ [F(f(x))] &= x + 1 > x = [F(x)] \end{aligned}$$

ensuring that the requirements in Theorem 5 are satisfied and then proving termination of  $\mathcal{R}$ .

---

<sup>4</sup>If we use the improvements in the definition of DP used in [HM04], the second DP does not appear.

The DPs can be presented as a *dependency graph*, where the infinite chains are represented by the *cycles* in the graph. In this dependency graph, we can decompose the problem of proving termination of a TRS into the problem of proving the absence of infinite chains of DPs which are part of the cycles in the graph. This modular decomposition permits the use of different orderings with different cycles [GAO02]. Further developments lead to more sophisticated frameworks for proving termination of rewriting:

- The *DP framework* [GTSK04, GTSKF06, Thi08], which is based on the *DP approach* [AG00, GAO02, HM04, HM05], transforms a proof of termination of a term rewriting system into a *DP problem*. The DP problems are transformed by using the so-called DP processors, which can be applied in a recursive way until trivial DP-problems are reached, which marks the end of the proof.
- A *constraint-based framework* [Bor03], which follows the idea of applying the monotonic semantic path ordering [BFR00] to produce a disjunction of ordering constraints, which characterise the termination of the TRS.

In the following, we provide a brief description of the DP Framework, which is important to be able to understand our development.

### 1.1.2 Dependency Pair Framework

The goal of the DP framework is to extend the DP approach to combine arbitrary termination techniques. In the DP framework, the origin of pairs and rules is relevant only in the initial problem. The signature and rules of the two sets can be disjoint.

The crucial feature of the DP framework when dealing with proofs of termination is to examine a set of pairs  $\mathcal{P}$ , which are intended to be (subsets of possibly transformed) DPs, together with the rules  $\mathcal{R}$  to prove the absence of (minimal) infinite  $(\mathcal{P}, \mathcal{R})$ -chains instead of just proving the absence of infinite rewrite sequences with  $\mathcal{R}$ . A  $(\mathcal{P}, \mathcal{R})$ -chain is a sequence  $u_1 \rightarrow v_1, u_2 \rightarrow v_2, \dots$  of pairs  $u_i \rightarrow v_i \in \mathcal{P}$  together with a substitution  $\sigma$  such that  $\sigma(v_i)$  rewrites to  $\sigma(u_{i+1})$  for all  $i \geq 1$ . With this idea, we can decompose a problem into several independent subproblems of similar structure. Again, these new problems can be solved independently by using different techniques, leading to a flexible and modular framework.

Roughly speaking, a *DP problem*  $(\mathcal{P}, \mathcal{R})$  consists of two TRSs  $\mathcal{P}$  and  $\mathcal{R}$ . The goal is to show that there is no infinite minimal  $(\mathcal{P}, \mathcal{R})$ -chain. A DP problem is called *finite* if there is no infinite minimal  $(\mathcal{P}, \mathcal{R})$ -chain; it is called

*infinite* if it is not finite or if  $\mathcal{R}$  is nonterminating. DP problems are transformed (and hopefully simplified) by means of the so-called DP processors. A *DP processor* is a function  $\text{Proc}$  that takes a DP problem and returns a (possibly empty) set of DP problems. Alternatively,  $\text{Proc}$  can return “no”. A DP processor is *sound* if for all DP problems  $\tau$ ,  $\tau$  is finite whenever  $\text{Proc}(\tau) \neq \text{“no”}$  and all DP problems in  $\text{Proc}(\tau)$  are finite. A DP processor is *complete* if for all DP problems  $\tau$ ,  $\tau$  is infinite whenever  $\text{Proc}(\tau) = \text{“no”}$  or  $\text{Proc}(\tau)$  contains an infinite DP problem. Soundness of  $\text{Proc}$  is required to prove termination. Completeness is needed to prove nontermination. The DP framework is formally introduced in the following theorem.

**Theorem 7 (DP Framework [GTSKF06, Thi08])** *Let  $\mathcal{R}$  be a TRS. We construct a tree whose nodes are labeled as DP problems, “yes”, or “no”, and whose root is labeled with  $(\text{DP}(\mathcal{R}), \mathcal{R})$ . For every inner node labeled with  $\tau$ , there is a sound processor  $\text{Proc}$  that satisfies one of the following conditions:*

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

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

Theorem 7 shows how the DP framework is used to (dis)prove termination of a TRS  $\mathcal{R}$ : we start with a DP problem  $(\text{DP}(\mathcal{R}), \mathcal{R})$  consisting of the DPs of  $\mathcal{R}$  and  $\mathcal{R}$  itself. Thus, DP processors are successively attempted until an empty set of DP problems is obtained (or, alternatively, a “no” label is returned) [GTSKF06].

## 1.2 Context-Sensitive Rewriting

The operational semantics of rewriting outlined in Section 1.1 looks easy: we just apply rules until a *normal* form is reached. However, sometimes several rules (even the same rule) can be applied to the same term. Therefore, we need to establish a (usually deterministic) *strategy* to decide which rule must be applied at any moment. In many cases, the termination behavior depends on the *rewriting strategy*. Roughly speaking, a *rewriting strategy* is a rule for appropriately choosing the rewriting steps to be issued in a computation.

Strategies help us to obtain an appropriate behavior in terms of efficiency, normalization, termination, etc. Eventually, this can create problems of efficiency or termination depending on the kind of strategy we use.

---

**Example 8**

Consider the following definition of the usual *if-then-else* rewriting rules:

$$\text{if}(\text{true}, x, y) \rightarrow x \quad (1.1)$$

$$\text{if}(\text{false}, x, y) \rightarrow y \quad (1.2)$$

This definition encodes the expected behavior of conditional expressions: depending on the outcome (true or false) of the evaluation of the first argument  $b$  in a call  $\text{if}(b, s, t)$ , we would evaluate the second ( $s$ ) or the third argument ( $t$ ) of the call.

However, in pure term rewriting, the three arguments  $b$ ,  $s$ , and  $t$  in the call could be evaluated in any order, thus eventually leading to wasteful computations (for instance, one could evaluate  $s$ ,  $t$ , and finally  $b$ !).

---

For this reason, the designers of programming languages have developed some features and language constructs that are aimed at giving the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to arguments of symbols) have been used in programming languages, such as Clean [NSEP91], Haskell [HPJW92], Lisp [McC60], Maude [CDE<sup>+</sup>07], OBJ2 [FGJM85], OBJ3 [GWM<sup>+</sup>00], CafeOBJ [FN97], etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become ‘more eager’ and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become ‘more lazy’, thus (hopefully) avoiding nontermination.

In term rewriting, these syntactic annotations are usually modeled by using *context-sensitive rewriting* [Luc98]. As we are going to see, context-sensitive rewriting (CSR [Luc98, Luc02]) provides a simple solution to the problem illustrated in Example 8. In CSR, besides the TRS  $\mathcal{R} = (\mathcal{F}, R)$ , we also consider a *replacement map*  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  that satisfies  $\mu(f) \subseteq \{1, \dots, k\}$ , for each  $k$ -ary symbol  $f$  in the signature  $\mathcal{F}$  [Luc98] (a pair  $(\mathcal{R}, \mu)$  is often called a CS-TRS). The replacement map  $\mu$  discriminates the argument positions  $\mu(f)$  of function symbols  $f$  where rewritings are allowed. A restriction of the rewriting computations is formalized at a very simple syntactic level: that of the arguments of function symbols  $f$  in the signature  $\mathcal{F}$ . In CSR we only rewrite  $\mu$ -replacing subterms: every term  $t$  (as a whole) is  $\mu$ -replacing by

definition; and  $t_i$  (as well as all its  $\mu$ -replacing subterms) is a  $\mu$ -replacing subterm of  $f(t_1, \dots, t_k)$  if  $i \in \mu(f)$ . In particular, if we fix  $\mu(\text{if}) = \{1\}$  for the TRS in Example 8, we avoid the aforementioned undesired computations. The following example provides a more illustrative case study involving conditional expressions as well.

---

**Example 9**

The following TRS  $\mathcal{R}$  [GM04, Example 49] provides a definition of integer division which advantageously uses CSR for handling *if-then-else* expressions:

$$\text{if}(\text{true}, x, y) \rightarrow x \quad (1.3)$$

$$\text{if}(\text{false}, x, y) \rightarrow y \quad (1.4)$$

$$0 \geq s(y) \rightarrow \text{false} \quad (1.5)$$

$$s(x) \geq s(y) \rightarrow x \geq y \quad (1.6)$$

$$x \geq 0 \rightarrow \text{true} \quad (1.7)$$

$$0 - y \rightarrow 0 \quad (1.8)$$

$$s(x) - s(y) \rightarrow x - y \quad (1.9)$$

$$0 \div s(y) \rightarrow 0 \quad (1.10)$$

$$s(x) \div s(y) \rightarrow \text{if}(x \geq y, s((x - y) \div s(y)), 0) \quad (1.11)$$

In this case, we want *if* to behave in such a way that we only evaluate the second and third arguments after the evaluation of the first argument. We can achieve this behavior with CSR by using a replacement map  $\mu$  such that  $\mu(\text{if}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{ar}(f)\}$  for all  $f \in \mathcal{F} \setminus \{\text{if}\}$ . By using the results in [Luc98], we could prove that, whenever a term  $t$  is rewritten into a normal form without *if* symbols, we can obtain the *same* normal form by using CSR (see [Luc98, Theorem 12]).

---

A sequence of rewriting steps that are performed on  $\mu$ -replacing subterms is called a  $\mu$ -rewrite sequence. CSR gives us the possibility to handle infinite structures with a terminating behavior for the context-sensitive rewrite relation. Given a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , we say that  $\mathcal{R}$  is  $\mu$ -terminating if no infinite  $\mu$ -rewrite sequence is possible.

---

**Example 10**

The TRS  $\mathcal{R}$  in Figure 1.1 can be used to compute approximations to  $\frac{\pi}{2}$  by using Wallis' product:

$$\frac{\pi}{2} = \lim_{n \rightarrow \infty} \frac{2}{1} \frac{2}{3} \frac{4}{5} \cdots \frac{2n}{2n-1} \frac{2n}{2n+1}$$

$$\begin{array}{lcl}
\text{evenNs} & \rightarrow & 0 : \text{incr}(\text{oddNs}) \\
\text{oddNs} & \rightarrow & \text{incr}(\text{evenNs}) \\
\text{incr}(x : xs) & \rightarrow & s(x) : \text{incr}(xs) \\
\text{take}(0, xs) & \rightarrow & [] \\
\text{take}(s(n), x : xs) & \rightarrow & x :: \text{take}(n, xs) \\
\text{zip}([], xs) & \rightarrow & [] \\
\text{zip}(xs, []) & \rightarrow & [] \\
\text{zip}(x : xs, y : ys) & \rightarrow & (x \div y) : \text{zip}(xs, ys) \\
\text{tail}(x : xs) & \rightarrow & xs \\
\text{rep2}([]) & \rightarrow & [] \\
\text{rep2}(x : xs) & \rightarrow & x : (x : \text{rep2}(xs)) \\
0 + n & \rightarrow & n \\
s(n) + m & \rightarrow & s(n + m) \\
0 * n & \rightarrow & 0 \\
s(n) * m & \rightarrow & m + (n * m) \\
\text{prodFrac}(x \div y, z \div t) & \rightarrow & (x * z) \div (y * t) \\
\text{prodOfFrac}([]) & \rightarrow & s(0) \div s(0) \\
\text{prodOfFrac}(p :: ps) & \rightarrow & \text{prodFrac}(p, \text{prodOfFrac}(ps)) \\
\text{halfPi}(n) & \rightarrow & \text{prodOfFrac}(\text{take}(n, \text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs}))))
\end{array}$$
Figure 1.1: Computing Wallis' approximation to  $\frac{\pi}{2}$ 

As usual, we represent (possibly infinite) lists as a recursive data structure where  $[]$  denotes the empty list, and  $x : xs$  is a list where  $x$  is the first element of the list and  $xs$  is a list that is called the *tail* of the list  $x : xs$ .

The evaluation of  $\text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs})))$  produces an infinite list containing the fractions in Wallis' formula. The function `take` can be used to obtain the components of a finite approximation to  $\frac{\pi}{2}$ , which we multiply with `prodOfFrac`. We collect such components in a *finite* list that is built by using a special (finite) list constructor  $(::)$ . A call `halfPi(sn(0))` for some positive number  $n > 0$  will return the desired approximation. Since  $\mathcal{R}$  is *nonterminating* (due to the first two rules), we should be careful when choosing the rewrite steps that will be issued to obtain an approximation.

With CSR, we can achieve a *terminating behavior* for this system. Consider the replacement map  $\mu$  given by:

$$\mu(\cdot) = \{1\} \text{ and } \mu(f) = \{1, \dots, \text{ar}(f)\} \text{ for all } f \in \mathcal{F} \setminus \{\cdot\}.$$

where  $\mu(\cdot) = \{1\}$  disallows reductions on the *list* part of the list constructor  $(\cdot)$ , thus making a kind of *lazy evaluation* of lists possible. Furthermore, the replacement restrictions imposed by the replacement map  $\mu$  are *not* an obstacle to obtain the desired approximations: the repeated application of CSR steps

to an expression  $\text{halfPi}(s^n(0))$  will obtain (disregarding the particular choice of such steps) an expression  $s^p(0) \div s^q(0)$  representing the approximation  $\frac{p}{q}$  to  $\frac{\pi}{2}$ , which is obtained by taking the first  $n$  terms in Wallis' formula (this follows from [Luc98, Theorem 11]).

---

It turns out that the use of CSR to *achieve* a terminating behavior with non-terminating TRSs has been considered as one of the most interesting aspects of CSR.

### 1.2.1 Applications of Termination of Context-Sensitive Rewriting

In [Luc06], a summary of applications of termination of CSR was presented. Roughly speaking, techniques for proving termination of CSR are useful to:

- **Prove Termination of Programs:** *Syntactic annotations* are useful for improving termination of programs in programming languages like CafeOBJ, OBJ, and Maude. Techniques for proving termination of CSR are often useful for proving termination of such programs.
- **Prove Termination of Rewriting:** Replacement maps can be used for relaxing monotonicity requirements that are imposed on the orderings that are used to prove termination of rewrite systems.
- **Prove Termination of Rewriting under Specific Strategies:** Termination of CSR has been used as a sufficient condition for termination of term rewriting systems using specific strategies or variants of rewriting:
  - Lazy rewriting [FKW00].
  - On-demand rewriting [Luc01].
  - Infinitary rewriting [KKS91, Mid97].
  - Innermost rewriting [Fer05].
- **Define Normalizing and Infinitary Normalizing Strategies:** Under some conditions, if termination of CSR is guaranteed for a TRS  $\mathcal{R}$ , then  $\mathcal{R}$  can be easily given normalizing and infinitary normalizing strategies [Luc02].
- **Prove Termination of Programmable Strategies:** Termination of CSR provides a sufficient condition for proving termination of strategies that are given as expressions of some specific strategy languages [FGK03].

Nowadays, there are several tools that can be used to prove termination of CSR. To our knowledge, they are AProVE [GSKT06], Jambox [End09], MU-TERM [Luc04b, AGIL07] and VMTL [SG09]. Therefore, state-of-the-art termination methods and automated termination provers for CSR become available for dealing with these applications.

Since 2006, new applications of termination of CSR have been discovered. In the following, we briefly enumerate some of them.

**Termination of Membership Equational Programs.** As we can see in [DLM<sup>+</sup>04, DLM<sup>+</sup>08], a sequence of theory transformations that can be used to bridge the gap between expressive membership equational programs [BJM00] and existing termination tools (which usually do not handle membership equational programs) is presented. These transformations take a membership equational program and obtain a context-sensitive rewrite system whose termination can be proved using existing termination tools.

**Outermost Termination of Rewriting.** In [EH09, EH10], the authors define a transformation from TRSs to context-sensitive TRSs in such a way that termination of the transformed CS-TRS implies outermost termination of the original system. For the class of left-linear<sup>5</sup> TRSs, the transformation is complete, i.e., for left-linear TRSs, termination of outermost rewriting is *characterized* as termination of CSR.

**Termination of Lazy Rewriting.** Lazy rewriting is a proper restriction of term rewriting that dynamically restricts the reduction of certain arguments of functions in order to obtain termination. In contrast to CSR, reductions at such argument positions are not completely forbidden but only delayed. In [SG08b], the authors develop a transformation from lazy rewrite systems into context-sensitive ones that is sound and complete with respect to (operational) termination<sup>6</sup>.

**Termination of Deterministic Conditional Term Rewriting Systems.** In [SG08a, SG10], the authors investigate termination of deterministic *conditional* term rewriting systems (DCTRSs), which is an important declarative programming paradigm where rewrite rules can be given equational *triggering* conditions. They show that operational termination of such systems can

<sup>5</sup>A TRS  $\mathcal{R}$  is left-linear if no variable occurs twice in  $\ell$  for all  $\ell \rightarrow r \in \mathcal{R}$ .

<sup>6</sup>See [LMM05] for more information and motivation about the notion of operational termination

be equivalently characterized as termination of a transformed CS-TRS *on the original terms* over the signature of the initial DCTRS.

**Termination with Negative Annotations.** In [AEGL10], the authors point out a number of problems of current proposals for handling on-demand strategy annotations, where negative numbers are used to indicate the arguments that should be evaluated only if this is necessary to match the left-hand side of a rule of the TRS. Then, they propose a solution to these problems by keeping an accurate track of annotations throughout the evaluation sequences. The on-demand evaluation strategy (ODE) overcomes the drawbacks of previous proposals and also has better computational properties. They also introduce a transformation for proving termination of the new evaluation strategy as termination of a transformed CS-TRS.

**Decidability of Termination of CSR.** In [USS08], the authors proved that termination of CSR is a decidable property for the class of semi-constructor term rewriting systems. This was the first formal result about decidability of termination of CSR.

**Termination of CSR with Built-In Numbers and Collection Data Structures.** In [FK09], the authors integrate replacement restrictions into the so-called constrained equational rewrite systems (CERSs [FK08]) and show how to prove termination of the obtained systems. Due to the benefits of CSR, they can manage infinite data structures as sets of integers avoiding infinite computations. This enables a more natural specification of some algorithms in the rewriting framework.

**Nontermination of CSR.** If a term rewrites to (an instance of) itself within a given context, then we have a loop. If a replacement map is considered, it is possible for this loop to occur in a frozen position. Otherwise, we have a context-sensitive loop. Being able to decide whether the loop is a context-sensitive loop is essential to obtain a non-trivial criterion for non- $\mu$ -termination of TRSs. A procedure to decide whether a loop is a context-sensitive loop has been developed in [TS09], which is the first work that addresses this problem.

**Proving Productivity in Infinite Data Structures** In [ZR10], the authors develop a general technique to prove productivity of specifications of infinite objects based on proving context-sensitive termination, and they present several examples.

### 1.2.2 Techniques for Proving Termination of CSR Automatically

In order to prove termination of CSR, two main approaches have been followed:

- *direct approaches*, that are based on the adaptation of rewriting termination techniques to CSR. For instance, generating  $\mu$ -reduction orderings [Zan97] that are compatible with the rules of the context-sensitive rewrite system (as in Lankford’s Theorem 2); and,
- *transformational approaches*, based on transforming a CS-TRS into a TRS in such a way that nontermination is preserved, i.e., if  $\mathcal{R}'$  is the transformed TRS which is obtained from a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , then  $\mathcal{R}$  is  $\mu$ -terminating whenever  $\mathcal{R}'$  is terminating.

The most popular direct approaches for proving termination of rewriting have been adapted to CSR: the context-sensitive recursive path ordering [BLR02], the polynomial orderings [GL02b, Luc04a, Luc05], the semantic path ordering [Bor03] and the Knuth-Bendix ordering [Bor03]. With respect to the transformational approach, several transformations from CS-TRSs into TRSs have been developed so far by Lucas [Luc96], Zantema [Zan97], Ferreira and Ribeiro [FR99] and Giesl and Middeldorp [GM99]. Comparative analyses about these transformations can be found in [GM04, Luc06].

Although there are a number of techniques for proving termination of CSR, the DP approach is the most powerful technique for proving termination of rewriting. This approach had not been investigated in connection with proofs of termination of CSR until 2006, when the research that is reported in this thesis started. In the following section, we summarize the contributions of this thesis.

## 1.3 Contributions of the Thesis

The main goal of this thesis is to improve the state-of-art of termination of CSR by providing a powerful and automatic framework for proving termination of CSR. The contributions of this thesis are summarized as follows:

1. A thorough analysis of infinite sequences in CSR. Starting with the notion of minimal nonterminating terms in standard rewriting [HM04], we adapt this notion and study how to obtain an appropriate notion for CSR.

2. Our analysis reveals the important role of *migrating variables* in the analysis of termination of CSR. The migrating variables of a rule are those variables that are frozen in the left-hand side of the rule and active in the right-hand side of the rule. We center our attention on the study of migrating variables, obtaining the notion of *hidden term* which is essential to model the non- $\mu$ -terminating behavior when migrating variables are present.
3. We obtain a sound and complete notion of context-sensitive dependency pair (CSDP) and CS dependency chain for CSR, where migrating variables generate dependency pairs of a new kind: the *collapsing CSDPs*, which are rules whose right-hand side is a variable.
4. By using our notions of CSDP and CS dependency chain, we extend the DP-framework to obtain a powerful CSDP-framework to automate the proofs of termination of CSR.
5. We adapt some of the most important processors used in rewriting to CSR: dependency graph, reduction pairs (with and without usable rules) or subterm criterion. In most cases, they are not direct adaptations from the unrestricted rewriting and require careful study. We also develop new processors using the advantages of having frozen positions to remove monotonicity requirements.
6. Finally, we have implemented all these improvements in a tool for proving termination properties, called MU-TERM.
7. Our experiments demonstrate that the results and techniques developed in this thesis are a valuable contribution that establishes a new state-of-art in the area.

## 1.4 Plan of the Thesis

This PhD thesis has been developed under the “publications format” that has recently been adopted for the presentation of PhD theses in Spanish universities<sup>7</sup>. According to this format a PhD can consist of a collection of papers previously published in relevant venues, together with an extended summary

---

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

of the research performed. In Part I, we present the summary itself. Part II contains the list of publications that develop the material in this thesis, accompanied by the full text for each publication as originally published.

The material in Part I is structured as follows:

1. In Chapter 2, we present some preliminary definitions and results, which are used in the main text of the thesis and also in the attached papers.
2. In Chapter 3, we investigate the structure of infinite context-sensitive rewrite sequences. This analysis is essential to obtain the notion of *hidden term*, which leads to identifying the structure of nonterminating  $\mu$ -rewrite sequences and, hence, to providing an appropriate definition of context-sensitive dependency pair (CSDP) and chain.
3. In Chapter 4, we introduce the notion of *context-sensitive dependency pair* and a corresponding notion of *chain*. The notion of CSDP chain is used to *characterize* termination of CSR.
4. In Chapter 5, we introduce a CSDP-framework that is easily mechanizable.
5. In Chapter 6, we describe several useful processors that can be used in the CSDP-framework to achieve proofs of termination of CSR.
6. In Chapter 7, we explain how this CSDP-framework is implemented in our tool MU-TERM.
7. In Chapter 8, we provide an experimental evaluation of our tool.
8. In Chapter 8.5, we present our conclusions and discuss some future work.

The publications collected in Part II are the following (in chronological order):

1. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 2006.
2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph**. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007. Selected papers from the 6th Spanish Conference on Programming and Languages, PROLE'06.

3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with MU-TERM**. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007. Selected papers from the 6th Spanish Conference on Programming and Languages, PROLE'06.
4. R. Gutiérrez, S. Lucas, and X. Urbain. **Usable Rules for Context-Sensitive Rewrite Systems**. In A. Voronkov, editor, *Proc. of XIX International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of *Lecture Notes in Computer Science*, pages 126–141, Hagenberg, Austria, 2008. Springer-Verlag.
5. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. **Improving Context-Sensitive Dependency Pairs**. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of XV International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *Lecture Notes in Computer Science*, pages 636–651. Springer-Verlag, 2008.
6. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. *Information and Computation*, 208:922–968, 2010.
7. R. Gutiérrez and S. Lucas. **Proving Termination in the Context-Sensitive Dependency Pairs Framework**. In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2010.
8. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. **Proving Termination Properties with MU-TERM**. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology And Software Technology, AMAST'10, Lecture Notes in Computer Science*, to appear, 2010. Springer-Verlag.



## Part I

# Summary of the Research



---

# 2

## Preliminaries

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

### 2.1 Abstract Reduction Systems

Let  $A$  be a set and  $R \subseteq A \times A$  be a binary relation on  $A$ . An *abstract reduction system* is a pair  $(A, R)$ . If  $a, b \in A$ , we write  $a R b$  and say that  $a$  *reduces to  $b$  in one step*, instead of  $(a, b) \in R$ . An *R-reduction sequence* is a finite or infinite sequence  $a_0 R a_1 R a_2 R a_3 R \dots$ . We denote the transitive closure of  $R$  by  $R^+$ , its reflexive closure by  $R^0$ , and its reflexive and transitive closure by  $R^*$ . An element  $a \in A$  is called an *R-normal form* if there exists no  $b$  such that  $a R b$ . We say that  $R$  is *terminating* (also known as *strongly normalizing*, *well-founded*, or *noetherian*) if there is no infinite reduction sequence  $a_1 R a_2 R a_3 \dots$ . A reflexive and transitive relation  $R$  is a quasi-ordering.

### 2.2 Signatures, Terms, and Positions

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

- $x \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  if  $x \in \mathcal{X}$ , and
- $f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  if  $t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $f \in \mathcal{F}$  and  $\text{ar}(f) = k$ .

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

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

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

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

### 2.3 Substitutions, Renamings, and Unifiers

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

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

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

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

## 2.4 Binary Relations over Terms

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

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

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

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

whenever  $s R t$ .

## 2.5 Rewrite Systems and Term Rewriting

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

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

## 2.6 Narrowing

*Narrowing* combines rewriting with unification. Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$ , a term  $s$  *narrows* to a term  $t$  (written  $s \xrightarrow{p}_{\mathcal{R}, \theta} t$ ,  $s \rightsquigarrow_{\mathcal{R}, \theta} t$ , or  $s \rightsquigarrow_{\theta} t$ ), if there is a nonvariable position  $p \in \mathcal{P}os_{\mathcal{F}}(s)$  and a rule  $\ell \rightarrow r \in R$  (sharing no variable with  $s$ ) such that  $s|_p$  and  $\ell$  unify with the most general unifier  $\theta$  and  $t = \theta(s[r]_p)$ .

## 2.7 Context-Sensitive Rewriting

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

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

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

whenever  $s R t$ .

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

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

**Example 11**

Consider  $\mathcal{R}$  and  $\mu$  as in Example 10. We have the following sequence (redexes are underlined):

$$\underline{\text{evenNs}} \hookrightarrow_{\mathcal{R},\mu} 0 : \text{incr}(\underline{\text{oddNs}}) \not\rightarrow_{\mathcal{R},\mu} 0 : \text{incr}(\text{incr}(\text{evenNs}))$$

Since the second argument of  $(:)$  is non- $\mu$ -replacing due to  $\mu(:) = \{1\}$ ,  $2.1 \notin \mathcal{P}os^\mu(0 : \text{incr}(\text{oddNs}))$ . Hence, the redex oddNs cannot be  $\mu$ -rewritten.

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

A term  $s$   $\mu$ -narrows to a term  $t$  (written  $s \overset{p}{\rightsquigarrow}_{\mathcal{R},\mu,\theta} t$ ,  $s \rightsquigarrow_{\mathcal{R},\mu,\theta} t$ , or  $s \rightsquigarrow_{\mu,\theta} t$ ), if  $s \overset{p}{\rightsquigarrow}_{\mathcal{R},\theta} t$  and  $p \in \mathcal{P}os_{\mathcal{F}}^\mu(s)$ .



---

# 3

## Structure of Infinite $\mu$ -Rewrite Sequences

Before 1987 [BL87], all methods for proving termination of rewriting were based on the idea of comparing the left- and right-hand sides of the rules by means of a suitable reduction ordering, i.e., a monotonic, stable, and well-founded ordering on terms (see [Der87] for a well-known survey of these methods). In 1996, Thomas Arts introduced the notion of DP [Art96], a notion that somewhat changed the way of thinking about termination. In [AG00], a sound and complete technique for proving termination of term rewriting based on the analysis of chains of DPs was presented. Intuitively, we can think about DPs as the representation of some possible (direct or indirect) recursive calls. Hirokawa and Middeldorp [HM04] made explicit a clear relation between infinite sequences and recursive function calls thanks to the notion of *minimal nonterminating term*. Given a TRS  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$ , the *minimal nonterminating terms* associated to  $\mathcal{R}$  are nonterminating terms  $t$  whose proper subterms  $u$  (i.e.,  $t \triangleright u$ ) are terminating. We denote as  $\mathcal{T}_\infty$  the set of minimal nonterminating terms associated to  $\mathcal{R}$  [HM04, HM07]. Considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term  $t = f(t_1, \dots, t_k) \in \mathcal{T}_\infty$ , is helpful to come to the notion of DP. Such sequences proceed as follows (see, e.g., [HM04]):

1. a finite number of reductions can be performed *below* the root of  $t$ , thus rewriting  $t$  into  $t'$ , which is *also minimal*, i.e.,  $t' \in \mathcal{T}_\infty$ ; then
2. a rule  $f(\ell_1, \dots, \ell_k) \rightarrow r$  applies *at the root* of  $t'$ , i.e.,  $t' = \sigma(f(\ell_1, \dots, \ell_k))$  for some substitution  $\sigma$ ; furthermore, since  $t' \in \mathcal{T}_\infty$ ,  $\sigma(x)$  is terminating for all  $x \in \text{Var}(f(\ell_1, \dots, \ell_k))$ ; and
3. there is a minimal nonterminating term  $u \in \mathcal{T}_\infty$  (hence,  $\text{root}(u) \in \mathcal{D}$ ) at some position  $p$  of  $\sigma(r)$  satisfying that  $p \in \text{Pos}_{\mathcal{F}}(r)$ , (i.e.,  $p$  is a

*nonvariable position of  $r$* ) that ‘continues’ the infinite sequence initiated by  $t$  in a similar way. The fact that  $\sigma(x)$  is terminating for all variables  $x$  in the rule is crucial for ensuring this important fact because it means that the contribution of some nonvariable part of the right-hand side  $r$  is essential for continuing the nonterminating rewrite sequence.

This means that considering the occurrences of defined symbols in the right-hand sides  $r$  of the rewrite rules suffices to ‘catch’ *every possible infinite rewrite sequence starting from  $\sigma(r)$* . In particular, no infinite sequence can be issued *below* the variables of  $r$  (more precisely: all bindings  $\sigma(x)$  are *terminating terms*).

In this chapter, we discuss how Hirokawa and Middeldorp’s results can be adapted to CSR. This study aims to obtain a novel and accurate notion of CSDP that models the behavior of infinite  $\mu$ -rewrite sequences in the next chapter.

### 3.1 Minimal Non- $\mu$ -Terminating Terms

Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$  and a replacement map  $\mu \in M_{\mathcal{F}}$ , a simple extension of the notion of minimal term for unrestricted rewriting introduced above (i.e.,  $\mathcal{T}_{\infty}$ ), is the following: let  $\mathcal{T}_{\infty, \mu}$  be a set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{T}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict subterm  $u$  of  $t$  (i.e.,  $t \triangleright u$ ) is  $\mu$ -terminating. Unfortunately, this definition fails when we try to reproduce the (essential) property (1) of infinite (context-sensitive) sequences starting from minimal terms.

#### Example 12

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

$$\begin{array}{lcl} f(c(x)) & \rightarrow & x \\ a & \rightarrow & c(f(a)) \end{array}$$

with  $\mu(f) = \{1\}$ ,  $\mu(c) = \mu(a) = \emptyset$ . Note that  $\mathcal{R}$  is not  $\mu$ -terminating because the following infinite  $\mu$ -rewrite sequence exists:

$$f(\underline{a}) \xrightarrow{\geq \Lambda}_{\mathcal{R}, \mu} \underline{f(c(f(a)))} \hookrightarrow_{\mathcal{R}, \mu} f(\underline{a}) \hookrightarrow_{\mathcal{R}, \mu} \dots$$

Note that  $f(\underline{a}) \xrightarrow{\geq \Lambda}_{\mathcal{R}, \mu} \underline{f(c(f(a)))}$  but  $f(c(f(a))) \notin \mathcal{T}_{\infty, \mu}$  because  $f(c(f(a))) \triangleright f(\underline{a})$  and  $f(\underline{a})$  is not  $\mu$ -terminating. Then, minimality with respect to  $\mathcal{T}_{\infty, \mu}$  is *not* preserved under inner  $\mu$ -rewritings.

Thus,  $\mathcal{T}_{\infty, \mu}$  is not appropriate for our purposes. In order to arrive to a suitable notion of minimality, we need to remove the requirement of having  $\mu$ -terminating terms at frozen positions:

**Definition 13 (Minimal Non- $\mu$ -Terminating Term [AGL06])** *Given a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , let  $\mathcal{M}_{\infty, \mu}$  be the set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict subterm  $u$  at an active position (i.e.,  $t \triangleright_{\mu} u$ ) is  $\mu$ -terminating.*

Note that  $\mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ . Now, we have (see [AGL10, Section 3.1]):

1. Every non- $\mu$ -terminating term  $s$  contains a minimal non- $\mu$ -terminating term  $t \in \mathcal{M}_{\infty, \mu}$  at an active position (i.e.,  $s \succeq_{\mu} t$ ), and
2. minimal non- $\mu$ -terminating terms  $t$  are always rooted by a *defined* symbol  $f \in \mathcal{D}$ :  $\forall t \in \mathcal{M}_{\infty, \mu}, \text{root}(t) \in \mathcal{D}$ .
3. Minimality is preserved under inner  $\mu$ -rewritings: for all  $s \in \mathcal{M}_{\infty, \mu}$ , if  $s \xrightarrow{\geq \Lambda}^* t$  and  $t$  is non- $\mu$ -terminating, then  $t \in \mathcal{M}_{\infty, \mu}$ .

Terms in  $\mathcal{T}_{\infty, \mu}$  are called *strongly minimal* non- $\mu$ -terminating terms and will also be used in Section 3.4.

---

**Example 14**

Continuing with Example 12, and the infinite sequence:

$$f(\underline{a}) \hookrightarrow_{\mathcal{R}, \mu} \underline{f(c(f(a)))} \hookrightarrow_{\mathcal{R}, \mu} f(\underline{a}) \hookrightarrow_{\mathcal{R}, \mu} \dots$$

we have that  $f(\underline{a}), \underline{f(c(f(a)))} \in \mathcal{M}_{\infty, \mu}$ , but only  $f(\underline{a}) \in \mathcal{T}_{\infty, \mu}$ .

---

## 3.2 Infinite $\mu$ -Rewrite Sequences Starting from Minimal Terms

Now we consider the structure of the infinite  $\mu$ -rewrite sequences starting from a minimal non- $\mu$ -terminating term  $t = f(t_1, \dots, t_k) \in \mathcal{M}_{\infty, \mu}$ . As summarized in Proposition 15 below, such sequences proceed as follows:

1. A finite number of reductions can be performed *below* the root of  $t$  at active positions, thus rewriting  $t$  into  $t'$ , which also belongs to  $\mathcal{M}_{\infty, \mu}$ , i.e.,  $t' \in \mathcal{M}_{\infty, \mu}$ ; then

2. a rule  $f(\ell_1, \dots, \ell_k) \rightarrow r$  applies *at the root* of  $t'$  (i.e.,  $t' = \sigma(f(\ell_1, \dots, \ell_k))$ ) for some substitution  $\sigma$  such that  $\sigma(x)$  is  $\mu$ -terminating for all  $\mu$ -replacing variables  $x \in \mathcal{V}ar^\mu(f(\ell_1, \dots, \ell_k))$ ; and
3. one of the following holds:
  - (a) There is a minimal non- $\mu$ -terminating term  $u \in \mathcal{M}_{\infty, \mu}$  at some position  $p$  of  $\sigma(r)$  satisfying that  $p \in \mathcal{P}os_F^\mu(r)$ , (i.e.,  $p$  is a *nonvariable and active position* of  $r$ ) that ‘continues’ the infinite sequence initiated by  $t$ .
  - (b) There is a *migrating variable*  $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(\ell)$  such that there is a minimal non- $\mu$ -terminating term  $u \in \mathcal{M}_{\infty, \mu}$  at some active position  $p$  of  $\sigma(x)$ . That is,  $\sigma(x) = C[u]_p$  for some context  $C[\ ]_p$  such that  $p \in \mathcal{P}os^\mu(C[\ ]_p)$ .

Item **3a** and Item **3b** arise as a consequence of relaxing the notion of minimal non- $\mu$ -terminating term: now, we know that  $\sigma(y)$  is terminating for all  $y \in \mathcal{V}ar^\mu(f(\ell_1, \dots, \ell_k))$ , but we do not know whether or not  $\sigma(x)$  is  $\mu$ -terminating when  $x \in \mathcal{V}ar(f(\ell_1, \dots, \ell_k)) \setminus \mathcal{V}ar^\mu(f(\ell_1, \dots, \ell_k))$ . This means that, in sharp contrast to the unrestricted case, considering the occurrences of defined symbols in the right-hand sides of the rewrite rules is *not* sufficient to ‘catch’ *every possible infinite rewrite sequence starting from  $\sigma(r)$* . We have to consider the *migrating variables* as well. These facts are summarized in the following important result.

**Proposition 15** [AGL10] *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then, for all  $t \in \mathcal{M}_{\infty, \mu}$ , there exist  $\ell \rightarrow r \in \mathcal{R}$ , a substitution  $\sigma$ , and a term  $u \in \mathcal{M}_{\infty, \mu}$  such that  $t \xrightarrow{\geq \Lambda}^* \sigma(\ell) \xrightarrow{\Lambda} \sigma(r) \triangleright_\mu u$  and either*

1. *there is a nonvariable  $\mu$ -replacing subterm  $s$  of  $r$ ,  $r \triangleright_\mu s$ , such that  $u = \sigma(s)$ , or*
2. *there is  $x \in \mathcal{V}ar^\mu(r) \setminus \mathcal{V}ar^\mu(\ell)$  such that  $\sigma(x) = C[u]_p$  for some context  $C[\ ]_p$  with  $p \in \mathcal{P}os^\mu(C[\ ]_p)$ .*

Proposition 15 entails the following result, which establishes some properties of infinite sequences starting from minimal non- $\mu$ -terminating terms.

**Corollary 16** [AGL10] *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . For all  $t \in \mathcal{M}_{\infty, \mu}$ , there is an infinite sequence*

$$t \xrightarrow{\geq \Lambda}^* \sigma_1(\ell_1) \xrightarrow{\Lambda} \sigma_1(r_1) \triangleright_\mu t_1 \xrightarrow{\geq \Lambda}^* \sigma_2(\ell_2) \xrightarrow{\Lambda} \sigma_2(r_2) \triangleright_\mu t_2 \xrightarrow{\geq \Lambda}^* \dots$$

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

1.  $t_i = \sigma_i(s_i)$  for some nonvariable term  $s_i$  such that  $r_i \supseteq_{\mu} s_i$ , or
2.  $\sigma_i(x_i) = C[t_i]_{p_i}$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$  and context  $C[\ ]_{p_i}$  such that  $p_i \in \mathcal{P}os^{\mu}(C_i[\ ]_{p_i})$ .

### 3.3 Hidden Terms

Our next goal is to thoroughly investigate Item 2 of Proposition 15 to obtain more information about the instantiation  $\sigma(x)$  of the migrating variable  $x$ .

One of the particularities of CSR that is absent in standard rewriting is that  $\mu$ -terminating terms can ‘generate’ non- $\mu$ -terminating subterms.

#### Example 17

Consider  $\mathcal{R}$  and  $\mu$  as in Example 12. We know that  $\mathbf{a}$  is  $\mu$ -terminating. If we apply the rule  $\mathbf{a} \rightarrow \mathbf{c}(\mathbf{f}(\mathbf{a}))$ , we obtain  $\mathbf{c}(\mathbf{f}(\mathbf{a}))$ , which contains the subterm  $\mathbf{f}(\mathbf{a})$ . Furthermore,  $\mathbf{f}(\mathbf{a})$  is non- $\mu$ -terminating.

Graphically, this situation is represented in Figure 3.1, where the grey zone represents a non- $\mu$ -replacing and non- $\mu$ -terminating (sub)term.

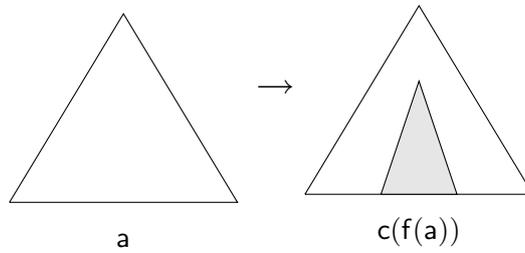


Figure 3.1: Delayed function call introduced by the rule  $\mathbf{a} \rightarrow \mathbf{c}(\mathbf{f}(\mathbf{a}))$

However, as stated in [AGL10, Lemma 1],  $\mu$ -termination is preserved under  $\mu$ -rewritings and extraction of  $\mu$ -replacing subterms.

**Lemma 18** [AGL10] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is  $\mu$ -terminating, then:*

1. *If  $s \supseteq_{\mu} t$ , then  $t$  is  $\mu$ -terminating.*

2. If  $s \hookrightarrow_{\mathcal{R}, \mu}^* t$ , then  $t$  is  $\mu$ -terminating.

This means that these non- $\mu$ -terminating terms introduced by  $\mu$ -rewriting steps can only occur at *frozen positions* in the reducts. The notion of *hidden term* captures these possible non- $\mu$ -terminating terms that occur at frozen positions in the right-hand sides of the rules in the TRS  $\mathcal{R}$ .

**Definition 19 (Hidden Terms [AEF<sup>+</sup>08, AGL10])** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . We say that  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$  is a hidden term if there is a rule  $\ell \rightarrow r \in R$  such that  $r \triangleright_{\mu} t$ . Let  $\mathcal{HT}(\mathcal{R}, \mu)$  (or just  $\mathcal{HT}$ , if  $\mathcal{R}$  and  $\mu$  are clear from the context) be the set of all hidden terms in  $(\mathcal{R}, \mu)$ .

---

**Example 20**

For  $\mathcal{R}$  and  $\mu$  as in Example 9, the hidden terms of  $\mathcal{R}$  are:  $s((x - y) \div s(y))$ ,  $x - y$ ,  $(x - y) \div s(y)$ ,  $s(y)$  and  $0$ , where  $x - y$  and  $(x - y) \div s(y)$  are rooted by defined symbols.

For  $\mathcal{R}$  and  $\mu$  as in Example 10, the hidden terms are  $\text{incr}(\text{oddNs})$ ,  $\text{oddNs}$ ,  $\text{incr}(xs)$ ,  $\text{zip}(xs, ys)$ ,  $x : \text{rep2}(xs)$  and  $\text{rep2}(xs)$ , where only  $x : \text{rep2}(xs)$  is not rooted by a defined symbol.

---

Non- $\mu$ -terminating terms at frozen positions can be activated by some specific contexts.

---

**Example 21**

Consider again  $\mathcal{R}$  and  $\mu$  as in Example 12, but instead of  $\mathbf{a}$ , consider  $f(\mathbf{a})$ , which is minimal and non- $\mu$ -terminating. If we apply the rule  $\mathbf{a} \rightarrow c(f(\mathbf{a}))$  to this term, we obtain  $f(c(f(\mathbf{a})))$ , which contains a context  $f(c(\square))$  that allows us to move the non- $\mu$ -replacing and non- $\mu$ -terminating term  $f(\mathbf{a})$  introduced by the rule to an active position by means of the rule  $f(c(x)) \rightarrow x$ , where  $x$  is a migrating variable.

Graphically, this behavior is represented in Figure 3.2, where the non- $\mu$ -terminating term introduced by the rule  $\mathbf{a} \rightarrow c(f(\mathbf{a}))$  in Figure 3.1 is activated by the migrating variable  $x$  in the rule  $f(c(x)) \rightarrow x$ . According to Proposition 15, the migrating variable  $x$  permits the activation of the non- $\mu$ -terminating term  $\sigma(x) = f(\mathbf{a})$ .

---

**Remark 22** The role of hidden terms in the description of the binding  $\sigma(x)$  for migrating variables  $x \in \text{Var}^{\mu}(r) \setminus \text{Var}^{\mu}(\ell)$  is going to be the following: assume  $x$ ,  $\sigma$ ,  $C[\ ]_p$  and  $u$  as in Proposition 15 (2); then  $\sigma(x) = C[u]_p$  and  $u = \theta(u')$  for some hidden term  $u'$  and substitution  $\theta$  (see Theorem 28 below). ■

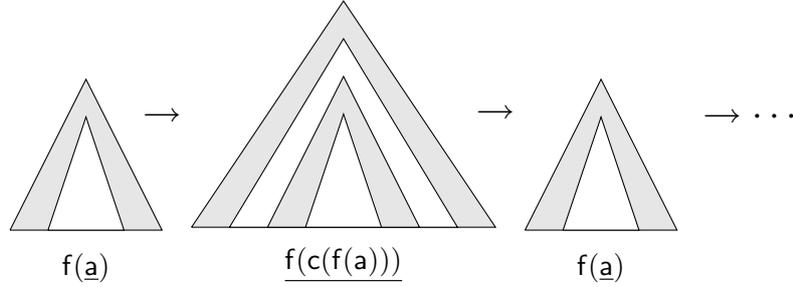


Figure 3.2: The delayed function call is activated by the application of the rule  $f(c(x)) \rightarrow x$

---

**Example 23**

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

$$\begin{array}{ll} \mathbf{a} \rightarrow \mathbf{f}(\mathbf{c}(\mathbf{b})) & \mathbf{h}(x) \rightarrow x \\ \mathbf{f}(x) \rightarrow \mathbf{h}(\mathbf{d}(x)) & \mathbf{b} \rightarrow \mathbf{a} \end{array}$$

together with  $\mu(\mathbf{c}) = \mu(\mathbf{d}) = \{1\}$  and  $\mu(\mathbf{a}) = \mu(\mathbf{b}) = \mu(\mathbf{f}) = \mu(\mathbf{h}) = \emptyset$ . The CS-TRS is non- $\mu$ -terminating due to the following  $\mu$ -rewrite sequence:

$$\underline{\mathbf{a}} \hookrightarrow_{\mathcal{R}, \mu} \underline{\mathbf{f}(\mathbf{c}(\mathbf{b}))} \hookrightarrow_{\mathcal{R}, \mu} \underline{\mathbf{h}(\mathbf{d}(\mathbf{c}(\mathbf{b})))} \hookrightarrow_{\mathcal{R}, \mu} \underline{\mathbf{d}(\mathbf{c}(\underline{\mathbf{b}}))} \hookrightarrow_{\mathcal{R}, \mu} \underline{\mathbf{d}(\mathbf{c}(\underline{\mathbf{a}}))} \hookrightarrow_{\mathcal{R}, \mu} \dots$$

The hidden terms in the system are  $\mathbf{c}(\mathbf{b})$ ,  $\mathbf{b}$ , and  $\mathbf{d}(x)$ . When the rule  $\mathbf{h}(x) \rightarrow x$  is applied in the third  $\mu$ -rewriting step, we have that  $\sigma(x) = C[u]_p = \mathbf{d}(\mathbf{c}(\mathbf{b}))$  where  $C[\square] = \mathbf{d}(\mathbf{c}(\square))$  and  $u = \mathbf{b}$ . Note that  $u$  is a hidden term.

---

We need to know more about the context  $C[\ ]_p$  in Proposition 15 (2) where the minimal term  $u$  occurs. Since the delayed function call  $u$  in Proposition 15 (2) is at a frozen position until its activation, the context  $C[\ ]_p$  can only be composed by symbols  $\mathbf{f}$  contained in hidden terms  $\mathbf{f}(\dots, r_i, \dots)$  such that  $r' \triangleright_{\mu} \mathbf{f}(\dots, r_i, \dots) \triangleright_{\mu} r_i$  for a rule  $\ell' \rightarrow r' \in \mathcal{R}$  which satisfies certain properties:

- $r_i$  is a nonvariable term and  $\sigma(r_i) = u$  (depicted in Figure 3.3 for Example 23), or
- $r_i$  is a variable at a frozen position in  $\ell$  and in  $r$  (depicted in Figure 3.4 for Example 23).

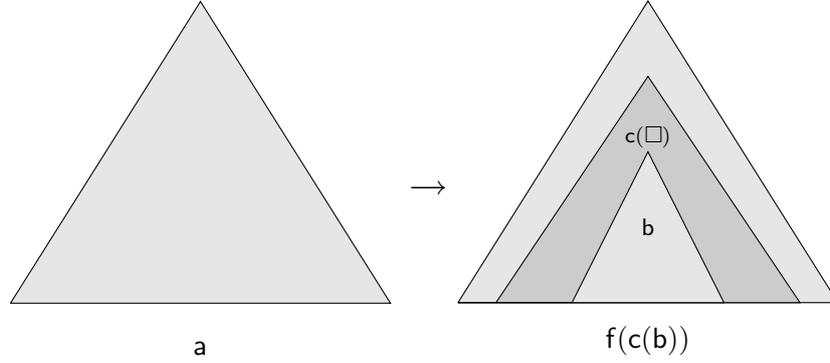


Figure 3.3: The delayed function call  $\mathbf{b}$  is introduced by the application of the rule  $\mathbf{a} \rightarrow \mathbf{f}(\mathbf{c}(\mathbf{b}))$

The symbols in this context conform the *hiding context*.

**Definition 24 (Hiding Context [GL10a])** Let  $\mathcal{R} = (\mathcal{F}, \mathcal{R}) = (\mathcal{C} \uplus \mathcal{D}, \mathcal{R})$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . A function symbol  $\mathbf{f} \in \mathcal{F}$  hides position  $i \in \mu(\mathbf{f})$  in the rule  $\ell \rightarrow r \in \mathcal{R}$  if  $r \triangleright_{\mu} \mathbf{f}(r_1, \dots, r_n)$  for some terms  $r_1, \dots, r_n$ , and  $r_i$  contains a  $\mu$ -replacing defined symbol (i.e.,  $\text{Pos}_{\mathcal{D}}^{\mu}(r_i) \neq \emptyset$ ) or a variable  $x \in (\text{Var}^{\mu}(\ell) \cap \text{Var}^{\mu}(r)) \setminus (\text{Var}^{\mu}(\ell) \cup \text{Var}^{\mu}(r))$  which is  $\mu$ -replacing in  $r_i$  (i.e.,  $x \in \text{Var}^{\mu}(r_i)$ ). We say that  $\mathbf{f}$  hides position  $i$  in  $\mathcal{R}$  if there is a rule  $\ell \rightarrow r \in \mathcal{R}$  such that  $\mathbf{f}$  hides position  $i$  in  $\ell \rightarrow r$ . A context  $C[\square]$  is hiding if

1.  $C[\square] = \square$ , or
2.  $C[\square] = \mathbf{f}(t_1, \dots, t_{i-1}, C'[\square], t_{i+1}, \dots, t_k)$ , where  $\mathbf{f}$  hides position  $i$  in  $\mathcal{R}$  and  $C'[\square]$  is a hiding context.

---

### Example 25

Continuing with the CS-TRS  $(\mathcal{R}, \mu)$  in Example 9, we have that symbol  $\mathbf{s}$  hides position 1, and symbol  $(\div)$  hides position 1.

And, continuing with the CS-TRS  $(\mathcal{R}, \mu)$  in Example 10, we have that symbol  $\mathbf{incr}$  hides position 1, symbol  $\mathbf{zip}$  hides positions 1 and 2, and symbol  $\mathbf{rep2}$  hides position 1.

---

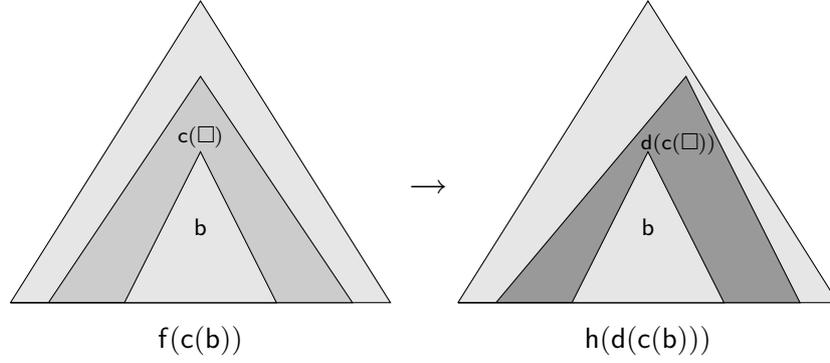


Figure 3.4: The context of the delayed function call  $b$  is modified by the application of the rule  $f(x) \rightarrow h(d(x))$

### 3.4 Infinite $\mu$ -Rewrite Sequences Starting from Strongly Minimal Terms

In the following, we consider a function  $\text{REN}^\mu$  which *independently* renames all *occurrences* of  $\mu$ -replacing variables within a term  $t$  by using fresh variables that are not in  $\text{Var}(t)$ :

**Definition 26** ( $\text{REN}^\mu$  [AGL10]) *Given a replacement map  $\mu$  over a signature  $\mathcal{F}$ , we let  $\text{REN}^\mu$  as follows:*

- $\text{REN}^\mu(x) = y$  if  $x$  is a variable, where  $y$  is intended to be a fresh variable that has not yet been used; and
- $\text{REN}^\mu(f(t_1, \dots, t_k)) = f([t_1]_1^f, \dots, [t_k]_k^f)$  for every  $k$ -ary symbol  $f \in \mathcal{F}$ , where given a term  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $[s]_i^f = \text{REN}^\mu(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ .

Note that  $\text{REN}^\mu(t)$  keeps variables at non- $\mu$ -replacing positions untouched. Note also that  $\text{REN}^\mu$  is *not* a substitution: it replaces the  $n(x)$  different  $\mu$ -replacing occurrences of the *same* variable  $x$  by *different* variables  $x_1, \dots, x_n$ .

Now, we are ready to combine the notions and results developed in this chapter into a main result establishing how infinite  $\mu$ -rewrite sequences starting from *strongly* minimal non- $\mu$ -terminating terms proceed. This can be done by assuming strong minimality for the first element  $t$  in Corollary 16.

The use of  $\text{REN}^\mu$  together with  $\mu$ -narrowability yields a necessary condition for reducibility of terms under some instantiations which is used in our development.

**Proposition 27** [AGL10] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$  be a nonvariable term and  $\sigma$  be a substitution. If  $\sigma(t) \xrightarrow{>\Lambda^*} \sigma(\ell)$  for some (possibly renamed) rule  $\ell \rightarrow r \in \mathcal{R}$ , then  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable.*

In the following result, we write  $\text{NARR}_{\mathcal{R}}^\mu(t)$  (or just  $\text{NARR}^\mu(t)$ ) to indicate that  $t$  is  $\mu$ -narrowable w.r.t. the (intended) TRS  $\mathcal{R}$ . We also let

$$\mathcal{NH}\mathcal{T}(\mathcal{R}, \mu) = \{t \in \mathcal{HT}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D} \text{ and } \text{NARR}_{\mathcal{R}}^\mu(\text{REN}^\mu(t))\}$$

be the set of *hidden terms* that are rooted by a *defined* symbol, and that after applying  $\text{REN}^\mu$  become  $\mu$ -narrowable.

**Theorem 28 (Minimal Sequence [GL10a])** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . For all  $t \in \mathcal{T}_{\infty, \mu}$ , there is an infinite sequence*

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

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

1.  $t_i = \sigma_i(s_i)$  for some  $s_i \notin \mathcal{X}$  such that  $r_i \supseteq_\mu s_i$  and  $\text{root}(s_i) \in \mathcal{D}$ , or
2.  $\sigma_i(x_i) = \theta_i(C_i[t'_i])$  and  $t_i = \theta_i(t'_i)$  for some variable  $x_i \in \text{Var}^\mu(r_i) \setminus \text{Var}^\mu(\ell_i)$ ,  $t'_i \in \mathcal{NH}\mathcal{T}(\mathcal{R}, \mu)$ , hiding context  $C_i[\square]$ , and substitution  $\theta_i$ .

In Chapter 4, we use these results and notions to define a suitable notion of CSDP.

### 3.5 Historical Development of Infinite $\mu$ -Rewrite Sequences

In [AGL06], we stated Proposition 15. According to this proposition, we know that we have to look into the instantiated migrating variable  $\sigma(x)$  by using the  $\mu$ -subterm relation  $\supseteq_\mu$  in order to find the next active function call  $u$ , i.e.,  $\sigma(x) \supseteq_\mu u$ . The only information about  $u$  that is provided by Proposition 15 is that  $\text{root}(u)$  is a defined symbol (due to the minimality of  $u$ ).

In [AGL07], we obtained the first intuition about our latter refinements, where instead of considering all the possible terms rooted by a defined symbol,

we could establish that  $u$  was rooted by a defined *hidden symbol*<sup>1</sup>. In [AGL10], we refined this definition considering hidden terms instead of terms rooted by hidden symbols, defining the notion of *strongly minimal non- $\mu$ -terminating terms*  $\mathcal{T}_{\infty, \mu}$  and assuming  $\mu$ -narrowability of  $\mathcal{DHT}$  thanks to [AGL10, Corollary 2].

In [AEF<sup>+</sup>08], we define the notion of hiding context by the first time.

**Definition 29 (Hiding Context [AEF<sup>+</sup>08])** *Let  $(\mathcal{R}, \mu)$  be a CS-TRS. The function symbol  $f$  hides position  $i$  if there is a rule  $\ell \rightarrow r \in \mathcal{R}$  with  $r \triangleright_{\mu} f(r_1, \dots, r_i, \dots, r_k)$ ,  $i \in \mu(f)$ , and  $r_i$  contains a defined symbol or a variable at an active position. A context  $C$  is hiding iff  $C = \square$  or  $C$  has the form  $f(t_1, \dots, t_{i-1}, C', t_{i+1}, \dots, t_k)$  where  $f$  hides position  $i$  and  $C'$  is a hiding context.*

This notion was refined in [GL10a] to its actual definition. In [GL10a], we also integrate it with the results in [AGL10] about structures of infinite  $\mu$ -rewrite sequences to obtain the definitive result in Theorem 28, which is a refinement of [AGL10, Theorem 1].

---

<sup>1</sup>a symbol  $f$  is hidden if  $\text{root}(s) = f$  for some hidden term  $s$ .



---

# 4

## Context-Sensitive Dependency Pairs

According to Theorem 28, an infinite minimal  $\mu$ -rewriting sequence whose starting term  $t$  is strongly minimal has the following form:

$$t = t_0 \xrightarrow{\geq \Lambda}^* \sigma_1(\ell_1) \xrightarrow{\Lambda}_{\mathcal{R}, \mu} \sigma_1(r_1) \succeq_{\mu} t_1 \xrightarrow{\geq \Lambda}^* \sigma_2(\ell_2) \xrightarrow{\Lambda}_{\mathcal{R}, \mu} \dots$$

where  $t_i$  are minimal non- $\mu$ -terminating terms, for all  $i \geq 1$ . Then, we proceed by first performing some  $\mu$ -rewriting steps *below the root* of  $t_{i-1}$  to obtain a term  $\sigma_i(\ell_i)$  (i.e.,  $t_{i-1} \xrightarrow{\geq \Lambda}^* \sigma_i(\ell_i)$ ) and then applying a rule  $\ell_i \rightarrow r_i$  at the *topmost* position for some matching substitution  $\sigma_i$ . The application of such a rule either

1. *introduces* a new minimal non- $\mu$ -terminating subterm  $t_i = \sigma_i(s_i)$ , where  $s_i$  is a  $\mu$ -replacing nonvariable subterm of  $r_i$  which is rooted by a defined symbol (i.e.,  $r_i \succeq_{\mu} s_i$  and  $\text{root}(s_i) \in \mathcal{D}$ ), or
2. *takes* a minimal non- $\mu$ -terminating and non- $\mu$ -replacing subterm  $t_i$  from  $\sigma_i(\ell_i)$  (i.e.,  $\sigma_i(\ell_i) \triangleright_{\mu} t_i$ ) and
  - (a) brings it up to an *active* position by means of the binding  $\sigma_i(x_i)$  for some *migrating variable*  $x_i$  in  $\ell_i \rightarrow r_i$ ,  $\sigma(x_i) = \theta(C_i[t'_i])$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$  and a context  $C_i[\square]$  with a  $\mu$ -replacing hole.
  - (b) At this point, we know that  $\sigma(x_i) = \theta(C_i[t'_i])$ , where  $t'_i$  is rooted by a defined symbol due to  $t_i \in \mathcal{M}_{\infty, \mu}$ . Furthermore,  $t_i$  is an instance of the  $\mu$ -narrowable hidden term  $t'_i \in \mathcal{NHT}$  and  $C_i[\square]$  is an instance of a hiding context.

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

## 4.1 Defining CSDPs

The notion of CSDP comes directly from Proposition 15 where we have to distinguish two kind of function calls: the *direct* function calls represented by terms  $u$  as in Proposition 15 (1) and the *delayed* function calls represented by terms  $u$  as in Proposition 15 (2). This proposition gives us the key to formalize the delayed function calls thanks to the notion of migrating variable. Migrating variables are ultimately the ones responsible for the activation of delayed function calls. Then, we have two sets of CSDPs:

- $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , which represents all possible *direct* ‘recursive’ calls in the very same sense of original DPs, and
- $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , which represents the activation of *delayed* function calls by migrating variables.

In the following definition, given a signature  $\mathcal{F}$  and  $f \in \mathcal{F}$ , we let  $f^\sharp$  be a fresh symbol (often called *tuple* symbol or DP-symbol) that is associated to a symbol  $f$  [AG00]. Let  $\mathcal{F}^\sharp$  be the set of tuple symbols associated to symbols in  $\mathcal{F}$ . As usual, for  $t = f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^\sharp$  to denote the *marked* term  $f^\sharp(t_1, \dots, t_k)$ .

**Definition 30 (Context-Sensitive Dependency Pairs [AGL10])** *Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  be the set of context-sensitive dependency pairs (CSDPs) where:*

$$\begin{aligned} \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \succeq_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\prec_\mu s, \text{NARR}^\mu(\text{REN}^\mu(s))\} \\ \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow x \mid \ell \rightarrow r \in R, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell)\} \end{aligned}$$

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

As in [HM04] (which follows Dershowitz’s proposal in [Der04]), we require that subterms  $s$  of the right-hand sides  $r$  of the rules  $\ell \rightarrow r$ , which are used to build the CSDPs  $\ell^\sharp \rightarrow s^\sharp$ , not be  $\mu$ -replacing subterms of the left-hand side (i.e.,  $\ell \not\prec_\mu s$ ). Also, as in [LM08], we require ‘ $\mu$ -narrowability’ of  $\text{REN}^\mu(s)$ :  $\text{NARR}^\mu(\text{REN}^\mu(s))$ , that remove pairs that cannot generate infinite sequences.

$$\begin{aligned}
s(n) +^\# m &\rightarrow n +^\# m & (4.1) \\
\text{HALFP1}(n) &\rightarrow \text{EVENNS} & (4.2) \\
\text{HALFP1}(n) &\rightarrow \text{ODDNS} & (4.3) \\
\text{HALFP1}(n) &\rightarrow \text{PRODOFFRACS}(\text{take}(n, \text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \\
&\quad \text{tail}(\text{rep2}(\text{oddNs})))) & (4.4) \\
\text{HALFP1}(n) &\rightarrow \text{REP2}(\text{oddNs}) & (4.5) \\
\text{HALFP1}(n) &\rightarrow \text{REP2}(\text{tail}(\text{evenNs})) & (4.6) \\
\text{HALFP1}(n) &\rightarrow \text{TAIL}(\text{evenNs}) & (4.7) \\
\text{HALFP1}(n) &\rightarrow \text{TAIL}(\text{rep2}(\text{oddNs})) & (4.8) \\
\text{HALFP1}(n) &\rightarrow \text{TAKE}(n, \text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs})))) & (4.9) \\
\text{HALFP1}(n) &\rightarrow \text{ZIP}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs}))) & (4.10) \\
\text{ODDNS} &\rightarrow \text{EVENNS} & (4.11) \\
\text{ODDNS} &\rightarrow \text{INCR}(\text{evenNs}) & (4.12) \\
s(n) *^\# m &\rightarrow m +^\# (n * m) & (4.13) \\
s(n) *^\# m &\rightarrow n *^\# m & (4.14) \\
\text{PRODFRAC}(x \div y, z \div t) &\rightarrow x *^\# z & (4.15) \\
\text{PRODFRAC}(x \div y, z \div t) &\rightarrow y *^\# t & (4.16) \\
\text{PRODOFFRACS}(p :: ps) &\rightarrow \text{PRODFRAC}(p, \text{prodOfFrac}(ps)) & (4.17) \\
\text{PRODOFFRACS}(p :: ps) &\rightarrow \text{PRODOFFRACS}(ps) & (4.18) \\
\text{TAKE}(s(n), x : xs) &\rightarrow \text{TAKE}(n, xs) & (4.19) \\
\text{TAIL}(x : xs) &\rightarrow xs & (4.20) \\
\text{TAKE}(s(n), x : xs) &\rightarrow xs & (4.21)
\end{aligned}$$

Figure 4.1: CSDPs of the CS-TRS in Example 10

This improvement can be seen as a necessary condition based on the study in Chapter 3 and Proposition 27. This improvement is also useful in practice, as we can see in [LM08, Example 17].

### Example 31

The set  $\text{DP}(\mathcal{R}, \mu)$  of CSDPs for  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mu$  as in Example 10 is given in Figure 4.1. All the marked symbols are represented in capital letters if possible (if not, they are represented as the same symbol with the mark  $\#$  at the end). We define  $\mu^\#$  as follows:

$$\begin{aligned}
\mu^\#(+^\#) &= \mu^\#(\text{TAKE}) = \mu^\#(\text{ZIP}) = \mu^\#(*^\#) = \mu^\#(\text{PRODFRAC}) = \{1, 2\} \\
\mu^\#(\text{HALFP1}) &= \mu^\#(\text{PRODOFFRACS}) = \mu^\#(\text{REP2}) = \mu^\#(\text{TAIL}) = \mu^\#(\text{INCR}) = \{1\} \\
\mu^\#(\text{EVENNS}) &= \mu^\#(\text{ODDNS}) = \emptyset \\
\mu^\#(f) &= \mu(f) \text{ for all } f \in \mathcal{F}
\end{aligned}$$

Note that we have two collapsing pairs: (4.20) and (4.21). From an operational point of view, (4.20) and (4.21) represent the activation (and possible evaluation) of the tail  $xs$  of a list  $x : xs$  under some conditions.

---

**Example 32**

With regard to the CS-TRS  $(\mathcal{R}, \mu)$  in Example 9, we have the following collapsing pairs that corresponds to the *if-then-else* rules (1.1) and (1.2):

$$\text{IF}(\text{true}, x, y) \rightarrow x \quad (4.22)$$

$$\text{IF}(\text{false}, x, y) \rightarrow y \quad (4.23)$$

These rules represent the activation of the second or third argument of IF if the first argument is evaluated to `true` or `false`, respectively. Furthermore, we have the following noncollapsing pairs:

$$s(x) \geq^\# s(y) \rightarrow x \geq^\# y \quad (4.24)$$

$$s(x) -^\# s(y) \rightarrow x -^\# y \quad (4.25)$$

$$s(x) \div^\# s(y) \rightarrow x \geq^\# y \quad (4.26)$$

$$s(x) \div^\# s(y) \rightarrow \text{IF}(x \geq y, s((x - y) \div s(y)), 0) \quad (4.27)$$

We define  $\mu^\#$  as:

$$\mu^\#(\geq^\#) = \mu^\#(-^\#) = \mu^\#(\div^\#) = \{1, 2\}$$

$$\mu^\#(\text{IF}) = \{1\}$$

$$\mu^\#(f) = \mu(f) \text{ for all } f \in \mathcal{F}$$

---

A rule  $\ell \rightarrow r$  of a TRS  $\mathcal{R}$  is  $\mu$ -conservative if  $\text{Var}^\mu(r) \subseteq \text{Var}^\mu(\ell)$ , i.e., there is no migrating variable;  $\mathcal{R}$  is  $\mu$ -conservative if all its rules are  $\mu$ -conservative (see [Luc96, Luc06]).

## 4.2 Unhiding TRS

An essential property of the DP approach is that it provides a *characterization* of termination of TRSs  $\mathcal{R}$  as the absence of infinite (minimal) *chains of DPs* [AG00, GTSKF06]. In the DP approach, a *chain of DPs* is a sequence  $u_i \rightarrow v_i$  of DPs together with a substitution  $\sigma$  such that  $\sigma(v_i)$  rewrites to  $\sigma(u_{i+1})$  for all  $i \geq 1$ . Regarding CSR and CSDPs, when considering CSDPs from  $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ ,

we proceed in a similar way to the standard case: we just change the rewritings by  $\mu$ -rewritings. This corresponds to Theorem 28 (1).

However, by Theorem 28 (2), when we consider a migrating variable  $x_i \in \mathcal{V}ar^\mu(r_i) \setminus \mathcal{V}ar^\mu(\ell_i)$ , we know that  $\sigma(x_i) = \theta_i(C_i[\bar{t}_i])$  where  $C_i[\ ]$  is a hiding context and  $\bar{t}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$  is a nonvariable hidden term such that  $\theta(\bar{t}_i)$   $\mu$ -rewrites to an instance  $\sigma(\ell_{i+1})$  of the left-hand side of another rule  $\ell_{i+1} \rightarrow r_{i+1}$ . Then, in CSR we have to add a new ingredient: the information corresponding to the hidden terms and the hiding context. This is necessary to obtain an accurate characterization of  $\mu$ -termination. We use an *unhiding TRS*  $\text{unh}(\mathcal{R}, \mu)$  to deal with this task (see Definition 33 below).

Roughly speaking, this unhiding TRS captures the situation described in the second item of Theorem 28. According to this, we have to:

1. remove the (instance of the) hiding context  $C_i[\ ]$  to extract the delayed call  $t_i$ , and then,
2. connect this delayed call, which is an instance  $\theta(\bar{t}_i)$  of a hidden term  $\bar{t}_i$ , and the next CSDP.

Therefore, the idea is to develop these two actions by using rewrite rules of two kinds:

- If  $\theta(C_i[\bar{t}_i]) = \theta(\mathbf{f}(t_1, \dots, t_{i-1}, C'_i[\bar{t}_i], t_{i+1}, \dots, t_k))$  then, since  $C_i[\ ]$  is a hiding context (Definition 24),  $\mathbf{f}$  hides position  $i$  and  $C'_i[\ ]$  is a hiding context as well. Then, we can extract  $\theta(C'_i[\bar{t}_i])$  from  $\theta(C_i[\bar{t}_i])$  by using the following *projection* rule:

$$\mathbf{f}(x_1, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_k) \rightarrow x_i$$

- Once  $t_i$  has been reached, we know that it is an instance  $t_i = \theta(\bar{t}_i)$  of a nonvariable hidden term  $\bar{t}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$  and we have to connect  $t_i$  with the next CSDP. Since the root of the CSDP is a marked symbol, we can do it by using a rule that just changes the root symbol by its *marked* version in the following way:

$$\bar{t}_i \rightarrow \bar{t}_i^\sharp$$

**Definition 33 (Unhiding TRS [GL10a])** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . We define  $\text{unh}(\mathcal{R}, \mu)$  as the TRS consisting of the following rules:

1.  $\mathbf{f}(x_1, \dots, x_i, \dots, x_k) \rightarrow x_i$  for all function symbols  $\mathbf{f}$  of arity  $k$ , distinct variables  $x_1, \dots, x_k$ , and  $1 \leq i \leq k$  such that  $\mathbf{f}$  hides position  $i$ , and

2.  $t \rightarrow t^\#$  for every  $t \in \mathcal{NHT}(\mathcal{R}, \mu)$ .

Then, the rules of the form  $f(x_1, \dots, x_k) \rightarrow x_i$  simulate the  $\mu$ -replacing sub-term relation for the hiding context and the rules  $t \rightarrow t^\#$  simulate the marking of root symbols.

---

**Example 34**

Continuing with the CS-TRS  $(\mathcal{R}, \mu)$  in Example 10, we only have to consider the following rules to obtain the unhiding TRS:

$$\begin{aligned} \text{evenNs} &\rightarrow 0 : \text{incr}(\text{oddNs}) \\ \text{incr}(x : xs) &\rightarrow s(x) : \text{incr}(xs) \\ \text{zip}(x : xs, y : ys) &\rightarrow (x \div y) : \text{zip}(xs, ys) \\ \text{rep2}(x : xs) &\rightarrow x : (x : \text{rep2}(xs)) \end{aligned}$$

we have that `incr`, `rep2` and `zip` hide position 1, and `zip` hides position 2. With this information, we define the first group of rules in Definition 33 to be:

$$\begin{aligned} \text{incr}(x) &\rightarrow x \\ \text{rep2}(x) &\rightarrow x \\ \text{zip}(x, y) &\rightarrow x \\ \text{zip}(x, y) &\rightarrow y \end{aligned}$$

We can also see that  $\mathcal{NHT}(\mathcal{R}, \mu)$  consists of the following terms: `incr(oddNs)`, `incr(xs)`, `oddNs`, `rep2(xs)` and `zip(xs, ys)`. We define the second group of rules in Definition 33 from these terms:

$$\begin{aligned} \text{incr}(\text{oddNs}) &\rightarrow \text{INCR}(\text{oddNs}) \\ \text{incr}(xs) &\rightarrow \text{INCR}(xs) \\ \text{oddNs} &\rightarrow \text{ODDNS} \\ \text{rep2}(xs) &\rightarrow \text{REP2}(xs) \\ \text{zip}(xs, ys) &\rightarrow \text{ZIP}(xs, ys) \end{aligned}$$

---

**Example 35**

Continuing with the CS-TRS  $(\mathcal{R}, \mu)$  in Example 9,  $\text{unh}(\mathcal{R}, \mu)$  is:

$$x \div y \rightarrow x \tag{4.28}$$

$$s(x) \rightarrow x \tag{4.29}$$

$$(x - y) \div s(y) \rightarrow (x - y) \div^\# s(y) \tag{4.30}$$

$$x - y \rightarrow x -^\# y \tag{4.31}$$


---

### 4.3 Chains of CSDPs

As in the DP approach, the termination of CS-TRSs  $(\mathcal{R}, \mu)$  can be characterized as the absence of infinite (minimal) *chains of CSDPs*. Essentially, a chain of DPs is a (finite or infinite) sequence  $u_i \rightarrow v_i$  of DPs  $u_i \rightarrow v_i \in \text{DP}(\mathcal{R})$  together with a substitution  $\sigma$  such that  $\sigma(v_i)$   $\mathcal{R}$ -rewrites into  $\sigma(u_{i+1})$  for all  $i \geq 1$ , i.e.,  $\sigma(v_i) \rightarrow_{\mathcal{R}}^* \sigma(u_{i+1})$  for all  $i \geq 1$  [AG00]. In the DP-framework [GTSK04, GTSKF06, Thi08] the origin of the pairs and the rules is made independent, but the notion of chain remains the same after replacing  $\text{DP}(\mathcal{R})$  by  $\mathcal{P}$ , where  $\mathcal{P}$  is an arbitrary TRS.

In order to adapt these ideas to CSR, the challenge now is to make the origin of hidden terms and hiding context independent from the notion of chain, as is done with the pairs  $\mathcal{P}$ . Then, in order to model the behavior of collapsing pairs, we introduce a new TRS  $\mathcal{S}$  which simulates the marking and the subterm relation by means of rewrite rules. In the following, given a TRS  $\mathcal{S}$ , we let  $\mathcal{S}_{\triangleright\mu}$  be the rules  $s \rightarrow t \in \mathcal{S}$  such that  $s \triangleright_{\mu} t$ ; and  $\mathcal{S}_{\#} = \mathcal{S} \setminus \mathcal{S}_{\triangleright\mu}$ .

**Remark 36** Note that, if we have an unhiding TRS  $\text{unh}(\mathcal{R}, \mu)$ , the rules in  $\text{unh}_{\triangleright\mu}(\mathcal{R}, \mu)$  correspond with the rules of the form  $f(x_1, \dots, x_i, \dots, x_k) \rightarrow x_i$  and the rules in  $\text{unh}_{\#}(\mathcal{R}, \mu)$  with the rules of the form  $t \rightarrow t^{\#}$ . ■

**Definition 37 (Chain of Pairs - Minimal Chain [GL10a])** 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  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma$  satisfying that, for all  $i \geq 1$ ,

1. if  $v_i \notin \text{Var}(u_i) \setminus \text{Var}^{\mu}(u_i)$ , then  $\sigma(v_i) = t_i \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , and
2. if  $v_i \in \text{Var}(u_i) \setminus \text{Var}^{\mu}(u_i)$ , then  $\sigma(v_i) \xrightarrow{\Lambda}_{\mathcal{S}_{\triangleright\mu}, \mu}^* \circ \xrightarrow{\Lambda}_{\mathcal{S}_{\#}, \mu} t_i \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ .

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.

The notions of CSDP and unhiding TRS give rise to a sound and complete characterization of termination of CSR where the set  $\mathcal{P}$  correspond to the CSDPs, the set  $\mathcal{R}$  to the rules of the system and the set  $\mathcal{S}$  corresponds to the unhiding TRS.

**Theorem 38 (Characterization of  $\mu$ -Termination [GL10a])** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\mathcal{P} = \text{DP}(\mathcal{R}, \mu)$  and  $\mathcal{S} = \text{unh}(\mathcal{R}, \mu)$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if there is no infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu^{\#})$ -chain.

## 4.4 Historical Development of CSDPs

The first attempt to develop a theory of dependency pairs for CSR started more than ten years ago when Salvador Lucas asked Thomas Arts (who was preparing the first presentation of the dependency pair method [Art97]) about the possibility of extending the dependency pair approach to CSR. Arts immediately noticed that the main problem of extending the existing results for ordinary rewriting to CSR was the possibility of having variables that are not replacing in the left-hand sides of the rules but that become replacing in the corresponding right-hand side. This is what we now call *migrating* variables. After this first failed attempt, the focus moved to transformations of CS-TRSs  $(\mathcal{R}, \mu)$  into ordinary TRSs  $\mathcal{R}_\Theta^\mu$  (where  $\Theta$  represents the transformation) in such a way that termination of  $\mathcal{R}_\Theta^\mu$  implies the  $\mu$ -termination of  $\mathcal{R}$  [GM04, Luc06].

During the spring of 2006, MU-TERM was under revision in preparation for its participation in the 2006 International Termination Competition. The idea of adapting DPs to CSR come up again. The first preliminary definition of CSDPs was the following:

**Definition 39 (First Preliminary Version of CSDPs)** *Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let*

$$\begin{aligned} \text{DP}_1(\mathcal{R}, \mu) &= \{ \ell^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \succeq_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\prec_\mu s \} \\ &\cup \{ \ell^\sharp \rightarrow \text{MUSUBTERM}(x) \mid \ell \rightarrow r \in R, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(l) \} \\ &\cup \{ \text{MUSUBTERM}(f(x_1, \dots, x_k)) \rightarrow \text{MUSUBTERM}(x_i) \mid f \in \mathcal{F}, i \in \mu(f) \} \\ &\cup \{ \text{MUSUBTERM}(f(x_1, \dots, x_k)) \rightarrow f^\sharp(x_1, \dots, x_k) \mid f \in \mathcal{D} \} \end{aligned}$$

with  $\mu^\sharp(f) = \mu(f)$  if  $f \in \mathcal{F}$ ,  $\mu^\sharp(f^\sharp) = \mu(f)$  if  $f \in \mathcal{D}$ , and  $\mu^\sharp(\text{MUSUBTERM}) = \emptyset$ .

Note that, following Arts and Giesl's definition, we did *not* yet consider collapsing pairs at that point.

The underlying idea was to handle migrating variables  $x$  by enclosing them inside a term  $\text{MUSUBTERM}(x)$  that (after instantiating  $x$  by means of a substitution  $\sigma$ ) would be able to start the search for a  $\mu$ -replacing subterm  $s = f(s_1, \dots, s_k)$  that (after marking its root symbol  $f$  as  $f^\sharp$ ) was able to connect with the left-hand side of the next CSDP in a chain.

At that moment, we did not have any knowledge about hidden terms and hiding contexts. Therefore, we had to take into account that all the symbols of the signature were hiding and that all the defined symbols in any subterm of an instantiated migrating variable were marked.

The notion of *chain* of CSDPs which was used here was essentially the standard one. All pairs were treated in the very same way and the only difference was that pairs were connected by using CSR instead of ordinary rewriting.

**Definition 40 (First Preliminary Version of Chain)** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Given  $\mathcal{P} \subseteq \text{DP}_1(\mathcal{R}, \mu)$ , an  $(\mathcal{P}, \mathcal{R}, \mu^\sharp)$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma$  satisfying that

$$\sigma(v_i) \hookrightarrow_{\mathcal{R}, \mu^\sharp}^* \sigma(u_{i+1})$$

for all  $i \geq 1$ .

The implementation of CSDPs according to Definitions 39 and 40 did not work very well in practice. The structure of pairs that dealt with migrating variables introduced many arcs in the corresponding graph and therefore many cycles. Thus, the following proposal was considered instead.

**Definition 41 (Second Preliminary Version of CSDPs)** Let  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\text{DP}_2(\mathcal{R}, \mu) = \text{DP}_{2, \mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{2, \mathcal{X}}(\mathcal{R}, \mu)$  where:

$$\begin{aligned} \text{DP}_{2, \mathcal{F}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \succeq_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\prec_\mu s\} \\ \text{DP}_{2, \mathcal{X}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow U_{\ell, f, x}(x) \mid \ell \rightarrow r \in R, f \in \mathcal{D}, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell)\} \\ &\cup \{U_{\ell, f, x}(f(x_1, \dots, x_k)) \rightarrow f^\sharp(x_1, \dots, x_k) \mid \ell \rightarrow r \in R, f \in \mathcal{D}\} \end{aligned}$$

and  $\mu^\sharp(f) = \mu(f)$  if  $f \in \mathcal{F}$ ,  $\mu^\sharp(f^\sharp) = \mu(f)$  if  $f \in \mathcal{D}$ , and  $\mu^\sharp(U_{\ell, f, x}) = \{1\}$  for all rules  $\ell \rightarrow r$ , symbols  $f$  and variables  $x$  originating one of such symbols.

Here, migrating variables  $x$  in a rule  $\ell \rightarrow r$  were enclosed inside a term  $U_{\ell, f, x}(x)$  that (after instantiating  $x$  by means of a substitution  $\sigma$ ) would be able to connect any  $\mu$ -replacing subterm  $s = f(s_1, \dots, s_k)$  of  $\sigma(x)$  (such that  $f$  is a defined symbol) with the left-hand side of the next CSDP in a sequence. Note that no explicit  $\mu$ -replacing subterm search is possible with this new definition of CSDP. Instead, this requirement was moved to the definition of chain. Now, although these CSDPs still remain as the ‘traditional’ ones, a clear distinction was made between two kinds of CSDPs: those that were obtained from the nonvariable parts of the right-hand sides of the rules ( $\text{DP}_{2, \mathcal{F}}(\mathcal{R}, \mu)$  in Definition 41) and those that were introduced for treating the migrating variables ( $\text{DP}_{2, \mathcal{X}}(\mathcal{R}, \mu)$  in Definition 41). Both kinds of CSDPs were clearly distinguished in the new definition of chain and the  $\mu$ -subterm requirement was used to describe how chains of such CSDPs are built.

**Definition 42 (Second Preliminary Version of Chain)** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Given  $\mathcal{P} \subseteq \text{DP}_2(\mathcal{R}, \mu)$ , an  $(\mathcal{P}, \mathcal{R}, \mu^\sharp)$ -chain is a sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$  such that there is a substitution  $\sigma$  satisfying either:

1.  $\sigma(v_i) \hookrightarrow_{\mathcal{R}, \mu^\sharp}^* \sigma(u_{i+1})$  if  $\text{root}(v_i) \neq U_{\ell, f, x}$ , or

2.  $v_i = U_{\ell, f, x}(x_i)$  and  $u_{i+1} = U_{\ell, f, x}(w_i)$  for some  $x_i \in \mathcal{X}$ ,  $w_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $\sigma(x_i) \succeq_{\mu} \sigma(w_i)$ .

for all  $i \geq 1$ .

A further evolution of this move led to the definition of CSDP and chain in [AGL06].

**Definition 43 (Context-Sensitive Dependency Pairs [AGL06])** Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS, and let  $\mu \in M_{\mathcal{F}}$ . Let  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  be the set of context-sensitive dependency pairs (CSDPs) where:

$$\begin{aligned} \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) &= \{\ell^{\sharp} \rightarrow s^{\sharp} \mid \ell \rightarrow r \in R, r \succeq_{\mu} s, \text{root}(s) \in \mathcal{D}, \ell \not\prec_{\mu} s, \} \\ \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) &= \{\ell^{\sharp} \rightarrow x \mid \ell \rightarrow r \in R, x \in \text{Var}^{\mu}(r) \setminus \text{Var}^{\mu}(\ell)\} \end{aligned}$$

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

Our theoretical and practical results led to considering collapsing CSDPs in the notion of CSDP and also led to the following notion of chain.

**Definition 44 (Chain of CSDPs in [AGL06])** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mu}$ . Given  $\mathcal{P} \subseteq \text{DP}(\mathcal{R}, \mu)$ , an  $(\mathcal{P}, \mathcal{R}, \mu^{\sharp})$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , such that there is a substitution  $\sigma$  satisfying both:

1.  $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
2. if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is  $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) \succeq_{\mu} s_i$  and  $s_i^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ .

for all  $i \geq 1$ .

Our current definition of CSDP (Definition 30) corresponds to [AGL10, Definition 4].

The notion of chain in Definition 44 was gradually improved with the notion of *hidden symbol* in [AGL07] and with the notion of *hidden term* in [AGL10]. The chain of symbols lying on positions above/on  $p \in \text{Pos}(t)$  is  $\text{prefix}_t(\Lambda) = \text{root}(t)$ ,  $\text{prefix}_t(i.p) = \text{root}(t).\text{prefix}_{t|_i}(p)$ . The strict prefix *sprefix* is  $\text{sprefix}_t(\Lambda) = \Lambda$ ,  $\text{sprefix}_t(p.i) = \text{prefix}_t(p)$ , i.e., the last symbol in  $\text{prefix}_t(p.i)$  is removed.

**Definition 45 (Chain of pairs - Minimal Chain [AGL10])** Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . A  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$  satisfying that, for all  $i \geq 1$ :

1. if  $v_i \notin \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , and
2. if  $v_i \in \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) = C_i[s_i]_{p_i}$  for some  $s_i$  and  $C_i[\ ]_{p_i}$  such that  $p_i \in \text{Pos}^\mu(C_i[\ ]_{p_i})$ ,  $\text{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ , and  $s_i^\sharp \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ .

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

1. if  $v_i \notin \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i)$  is  $(\mathcal{R}, \mu)$ -terminating, and
2. if  $v_i \in \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $s_i^\sharp$  is  $(\mathcal{R}, \mu)$ -terminating and  $\exists \bar{s}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$  such that  $s_i = \sigma(\bar{s}_i)$ .

In [AEF<sup>+</sup>08], we were tempted to remove collapsing CSDPs by using a transformation of collapsing pairs into ‘ordinary’ (i.e., noncollapsing) pairs. In this way, we could return to a notion of chain that is similar to the one given for ordinary rewriting.

**Definition 46** [AEF<sup>+</sup>08, Definition 9] *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \neq \emptyset$ , we introduce a fresh un hiding tuple symbol  $\text{U}$  and the following un hiding DPs:*

- $s \rightarrow \text{U}(x)$  for every  $s \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ ,
- $\text{U}(f(x_1, \dots, x_i, \dots, x_k)) \rightarrow \text{U}(x_i)$  for every function symbol  $f$  of any arity  $k$  and every  $1 \leq i \leq k$  where  $f$  hides position  $i$ , and
- $\text{U}(t) \rightarrow t^\sharp$  for every hidden term  $t$ .

Let  $\text{DP}_u(\mathcal{R}, \mu)$  be the set of all un hiding DPs (where  $\text{DP}_u(\mathcal{R}, \mu) = \emptyset$  whenever  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$ ). Then  $\text{DP}'(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_u(\mathcal{R}, \mu)$ .

The corresponding definition of *chain* is, essentially, Definition 40 (but see [AEF<sup>+</sup>08, Definition 11]).

#### Example 47

Following Definition 46 in Example 31, instead of the two collapsing pairs (4.20) and (4.21), we have to add the following CSDPs:

$$\text{TAIL}(x : xs) \rightarrow \text{U}(xs) \quad (4.32)$$

$$\text{TAKE}(s(n), x : xs) \rightarrow \text{U}(xs) \quad (4.33)$$

$$\text{U}(\text{incr}(x)) \rightarrow \text{U}(x) \quad (4.34)$$

$$\text{U}(\text{incr}(\text{oddNs})) \rightarrow \text{INCR}(\text{oddNs}) \quad (4.35)$$

$$\text{U}(\text{oddNs}) \rightarrow \text{ODDNS} \quad (4.36)$$

$$U(\text{rep2}(x)) \rightarrow U(x) \quad (4.37)$$

$$U(\text{zip}(x, y)) \rightarrow U(x) \quad (4.38)$$

$$U(\text{zip}(x, y)) \rightarrow U(y) \quad (4.39)$$

$$U(x : y) \rightarrow U(x) \quad (4.40)$$

However, removing collapsing pairs leads to a less accurate computational model of infinite  $\mu$ -rewrite sequences. As shown in Chapter 3, migrating variables and collapsing pairs (which are their counterpart as CSDPs) are an essential part of the theoretical description of termination of CSR. Hence, Definition 46 (and this could also be said of Definition 39 and Definition 42) leads to a less intuitive notion of CSDPs that directly affects the development of new techniques.

Therefore, in [GL10a], we emphasize that collapsing pairs should not be removed from the definition of CSDP. Instead, we use the new notion of *unhiding TRS* and maintain the benefits of collapsing pairs, thus, leading to our current Definition 37, which corresponds to [GL10a, Definition 5].

**Remark 48** Note that if rules  $f(x_1, \dots, x_k) \rightarrow x_i$  for all  $f \in \mathcal{D}$  and  $i \in \mu(f)$  (where  $x_1, \dots, x_k$  are variables) are used in  $\text{unh}_{\triangleright_\mu}(\mathcal{R}, \mu)$ , then we have the notion of minimal chain in [AGL10]; and if, additionally, rules  $f(x_1, \dots, x_k) \rightarrow F(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$  are used in  $\text{unh}_{\#}(\mathcal{R}, \mu)$ , then we have the original notion of chain from [AGL06]. ■

---

# 5

## Context-Sensitive Dependency Pair Framework

During the last ten years, the DP approach has evolved into a powerful framework for proving termination of TRSs in practice. From the already classical article by Arts and Giesl [AG00] to later developments corresponding to the so-called *DP framework* [GTSK04, GTSKF06, Thi08] many new improvements have been introduced. The DP framework provides a basis for mechanizing proofs of termination of TRSs using DPs. In this chapter, we adapt it to CSR.

With respect to termination proofs, the central notion now is that of *CS problem*: the goal is checking its finiteness or infinity. A (sound) processor basically transforms termination problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original termination problem implies the existence of an infinite chain in the transformed one. Here ‘simpler’ usually means that fewer elements are involved in the problems. Often, processors are not only sound but also *complete*. This is essential if we are interested in *disproving* termination.

**Definition 49 (CS Problem [GL10a])** A CS problem  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  is a tuple, 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 minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain. The CS problem  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  is infinite if it is not finite or if  $\mathcal{R}$  is non- $\mu$ -terminating.

We can apply different termination techniques such as CS processors to these CS problems.

**Definition 50 (CS Processor [AEF<sup>+</sup>08, AGL10])** A CS Processor is a function Proc that takes a CS problem as an input and returns a new set of CS problems. Alternatively, Proc can return “no”.

A CS processor Proc is sound if for all CS problems  $\tau$ ,  $\tau$  is finite whenever  $\text{Proc}(\tau) \neq \text{“no”}$  and  $\forall \tau' \in \text{Proc}(\tau)$ ,  $\tau'$  is finite. A CS processor Proc is

complete if for all CS problems  $\tau$ ,  $\tau$  is infinite whenever  $\text{Proc}(\tau) = \text{"no"}$  or there is  $\tau' \in \text{Proc}(\tau)$  such that  $\tau'$  is infinite.

Sound (and possibly complete) processors are used in a pipe (more precisely, a *strategy tree*) to incrementally simplify the original problem as much as possible, possibly decomposing it into smaller pieces that are then independently treated in the very same way. The trivial case of this process comes when finiteness of the CS problems  $\tau$  becomes obvious (e.g., if  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  and  $\mathcal{P}$  is empty). Then, we can provide a *positive* answer “yes” to the termination problem that is propagated upwards to the original problem in the root of the tree. In some cases it is also possible to witness *infinity* of a given termination problem; then a *negative* answer “no” can be provided and propagated upwards provided that all considered processors were *complete*. Of course, the termination problems that we treat here are undecidable (in general), thus “don’t know” answers can also be generated (for instance, by a timeout system that interrupts the usually complex search processes that are involved in the proofs). When all the answers are collected, a final conclusion about the whole termination problem can be given:

1. If we have positive answers (“yes”) for all the problems in the leaves of the tree, then we conclude “yes” as well;
2. If a negative answer (“no”) was raised somewhere and the processors that were used in the path from the root to the node producing the negative answer were *complete*, then we conclude “no” as well;
3. Otherwise, the conclusion is “don’t know”.

**Theorem 51 (CSDP Framework [GL10a])** *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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\#})$ . For every inner node labeled with  $\tau$ , there is a sound processor  $\text{Proc}$  that satisfies one of the following conditions:*

1.  $\text{Proc}(\tau) = \text{no}$  and the node has just one child that is labeled with “no”.
2.  $\text{Proc}(\tau) = \emptyset$  and the node has just one child that is labeled with “yes”.
3.  $\text{Proc}(\tau) \neq \text{no}$ ,  $\text{Proc}(\tau) \neq \emptyset$ , and the children of the node are labeled with the CS problems in  $\text{Proc}(\tau)$ .

*If all leaves of the tree are labeled with “yes”, then  $\mathcal{R}$  is  $\mu$ -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- $\mu$ -terminating.*

**Example 52**

Continuing with Example 10, in order to prove termination of  $(\mathcal{R}, \mu)$ , we start with the following CS problem:

$$\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$$

where  $\text{DP}(\mathcal{R}, \mu)$  is obtained in Example 31 and  $\text{unh}(\mathcal{R}, \mu)$  is obtained in Example 34.

The notions of graph, cycles, SCCs, etc., are also part of the framework, but (1) they are incorporated as *CS processors* now, and (2) they refer to the elements in the (different) problems that are obtained through the process outlined above. In Chapter 6, we enumerate several CS processors that have been implemented and used to prove termination of CSR automatically.

## 5.1 Historical Development of the CSDP Framework

In [GLU08], the notion of usable rule is given for an arbitrary set  $\mathcal{P}$  of pairs. However, in this work we did not provide a real definition of CSDP framework. In [AEF<sup>+</sup>08, AGL10], we gave the first definitions of CS problems and CS processors.

**Definition 53 (CS Problem [AEF<sup>+</sup>08, AGL10])** *A CS problem  $\tau$  is a tuple  $\tau = (\mathcal{P}, \mathcal{R}, \mu)$ , where  $\mathcal{R}$  and  $\mathcal{P}$  are TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . The CS problem  $\tau$  is finite if there is no infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain. The CS problem  $\tau$  is infinite if  $\mathcal{R}$  is non- $\mu$ -terminating or there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain.*

The problem with this framework in [AGL10] is that the notion of chain is highly dependent on the notion of hidden term (see Definition 45). In [AEF<sup>+</sup>08], this restriction does not occur in the framework, but the notion of collapsing pair gets lost. Of course, since the notions of chain in [AEF<sup>+</sup>08] and in [AGL10] differ, the corresponding processors are not exactly equivalent.

Our current definition of chain (Definition 37) considers a new TRS,  $\mathcal{S}$ , which becomes visible in the notion of CS problem. From a mechanical point of view, the differences among the three CSDP frameworks are:

- In [AGL10], the subterm condition and the marking are part of the notion of chain and parametrized by  $\mathcal{R}$ .

- In [AEF<sup>+</sup>08], the subterm condition (which is captured by the notion of *hide*) and the marking are encoded by means of rules in  $\mathcal{P}$ .
- In [GL10a], the subterm condition and the marking are encoded by the unhiding TRS and are explicitly isolated by means of the rules in  $\mathcal{S}$ . This decomposition allows us not only to treat these rules as pairs (if necessary), but also to create special processors that are related to  $\mathcal{S}$  only.

---

# 6

## CS Processors

In the following sections, we describe a number of sound and (most of them) complete CS processors.

### 6.1 Preprocessing for Rule Removal

A reduction pair  $(\succsim, \sqsupset)$  consists of a stable and monotonic quasi-ordering  $\succsim$ , and a stable and well-founded ordering  $\sqsupset$  satisfying either  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$  [KNT99]. Reduction pairs are used in the DP approach to witness the absence of infinite chains of (dependency) pairs by finding a *reduction pair*  $(\succsim, \sqsupset)$  that is compatible with the rules and the DPs:  $\ell \succsim r$  for all rewrite rules  $\ell \rightarrow r$  and  $u \sqsupset v$  for all DPs  $u \rightarrow v$ . In the DP framework [GTSK04, GTSKF06, Thi08] (but also in [GAO02, HM04, HM05, HM07]), they are used to obtain *smaller* sets of pairs  $\mathcal{P}' \subset \mathcal{P}$  by removing the *strict* pairs, i.e., those pairs  $u \rightarrow v \in \mathcal{P}$  such that  $u \sqsupset v$  (in this case, all other pairs  $u \rightarrow v$  that are not strict must be compatible with the quasi-ordering  $\succsim$ , i.e.,  $u \succsim v$  must hold).

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

Monotonicity is required for  $\succsim$  to deal with the application of rules  $\ell \rightarrow r$  to an arbitrary depth in terms. Since the pairs are ‘applied’ only at the root level, no monotonicity is required for  $\sqsupset$  (but, for this reason, we cannot compare the rules in  $\mathcal{R}$  using  $\sqsupset$ ). Endrullis et al. noted that *transitivity* is not necessary for the strict component  $\sqsupset$  because it is somehow ‘simulated’ by the compatibility requirement above [EWZ08].

In our setting, since we are interested in  $\mu$ -rewriting steps only, we can

relax the *monotonicity* requirements as follows.

**Definition 54 ( $\mu$ -Reduction Pair [AGL06])** Let  $\mathcal{F}$  be a signature and  $\mu \in M_{\mathcal{F}}$ . A  $\mu$ -reduction pair  $(\succsim, \sqsupset)$  consists of a stable and  $\mu$ -monotonic quasi-ordering  $\succsim$  and a well-founded stable relation  $\sqsupset$  on terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  which are compatible, i.e.,  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$ .

We say that  $(\succsim, \sqsupset)$  is  $\mu$ -monotonic if  $\sqsupset$  is  $\mu$ -monotonic.

The following result allows us to use a  $\mu$ -monotonic  $\mu$ -reduction pair as a preprocessing that removes some rewrite rules from the original rewrite system  $\mathcal{R}$  before starting a termination proof.

**Proposition 55 (Removing Strict Rewrite Rules [AGL10])** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $(\succsim, \sqsupset)$  be a  $\mu$ -monotonic  $\mu$ -reduction pair such that  $\ell(\succsim \cup \sqsupset) r$  for all  $\ell \rightarrow r \in R$ . Let  $\mathcal{R}_{\sqsupset} = \{\ell \rightarrow r \in R \mid \ell \sqsupset r\}$  and  $\mathcal{S} = \mathcal{R} \setminus \mathcal{R}_{\sqsupset}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if  $\mathcal{S}$  is  $\mu$ -terminating.

### Example 56

If we start with the CS-TRS  $(\mathcal{R}_0, \mu)$  where  $\mathcal{R}_0$  is  $\mathcal{R}$  in Example 9 and  $\mu$  is the replacement map in [GM04] (a much more restricted replacement map than the one used in our leading example):

$$\begin{aligned} \mu(\text{if}) &= \mu(\div) = \mu(\text{s}) = \{1\} \text{ and} \\ \mu(-) &= \mu(\geq) = \mu(0) = \mu(\text{true}) = \mu(\text{false}) = \emptyset \end{aligned}$$

applying Proposition 55, we obtain  $\mathcal{R}_1 = \mathcal{R}_0 \setminus \{(1.6), (1.10)\}$  thanks to the following polynomial interpretation:

$$\begin{array}{ll} [\div](x, y) = x + y + 1 & [\geq](x, y) = x \\ [\text{if}](x, y, z) = x + y + z & [-](x, y) = 0 \\ [0] = 0 & [\text{s}](x) = x + 1 \\ [\text{false}] = 0 & [\text{true}] = 0 \end{array}$$

We can apply Proposition 55 recursively. In this case, we can apply it once again to obtain  $(\mathcal{R}_2, \mu)$  where  $\mathcal{R}_2 = \mathcal{R}_1 \setminus \{(1.9)\}$ ; the rule (1.9) can be removed from  $\mathcal{R}_1$  by using the following polynomial interpretation:

$$\begin{array}{ll} [\div](x, y) = x + y & [\geq](x, y) = 0 \\ [\text{if}](x, y, z) = x + y + z & [-](x, y) = x \\ [0] = 0 & [\text{s}](x) = x + 1 \\ [\text{false}] = 0 & [\text{true}] = 0 \end{array}$$

In this way, termination of  $\mathcal{R}$  is equivalent to finiteness of the following CS problem

$$\tau = \{(\{(4.22), (4.23), (4.26), (4.27)\}, \mathcal{R}_2, \text{unh}(\mathcal{R}_2, \mu), \mu^\sharp)\}$$

where  $\text{unh}(\mathcal{R}_2, \mu) = \text{unh}(\mathcal{R}_0, \mu)$  is given in Example 35. Thanks to this pre-processing, finiteness of  $\tau$  is equivalent to termination of  $\mathcal{R}$ , but pairs (4.24) and (4.25) are not part of  $\tau$ .

## 6.2 Context-Sensitive (Dependency) Graph

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

In the DP approach [AG00], a *dependency graph*  $\text{DG}(\mathcal{R})$  is associated to the considered TRS  $\mathcal{R}$ . The nodes of the dependency graph are the DPs in  $\text{DP}(\mathcal{R})$ ; there is an arc from a DP  $u \rightarrow v$  to a DP  $u' \rightarrow v'$  if the pairs define a chain for some substitution  $\sigma$ .

In more recent approaches, the analysis of infinite chains of DPs as such is just a starting point. Very often, chains of DPs are *transformed* into chains of more general *pairs* that can no longer be considered DPs. This is the case for the *narrowing* or *instantiation* transformations, among others (for example, see [GTSKF06]). Still, the analysis of the cycles in the graph built from such pairs is useful for investigating the existence of infinite (minimal) chains of pairs. Thus, a more general notion of *graph of pairs*  $\text{DG}(\mathcal{P}, \mathcal{R})$  associated to a set of pairs  $\mathcal{P}$  and a TRS  $\mathcal{R}$  is considered; the pairs in  $\mathcal{P}$  are now used as the nodes of the graph, but they are connected by  $\mathcal{R}$ -rewriting in the same way [GTSKF06, Definition 7].

In the following section, we take into account these points to provide an appropriate definition of context-sensitive (dependency) graph.

### 6.2.1 Definition of the Context-Sensitive Graph

According to the discussion above, our starting point are three TRSs ( $\mathcal{R}$ ,  $\mathcal{P}$ , and  $\mathcal{S}$ ) together with a replacement map  $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$ . Our aim is to obtain a notion of graph that can represent all infinite chains of pairs. We can construct a graph where the nodes represent the pairs and the arcs represent the possible chains between pairs.

**Definition 57 (Context-Sensitive Graph of Pairs [GL10a])** *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  $G(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  has  $\mathcal{P}$  as the set of nodes. Given  $u \rightarrow v, u' \rightarrow v' \in \mathcal{P}$ , there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$ , if  $u \rightarrow v, u' \rightarrow v'$  is a minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain for some substitution  $\sigma$ .*

With these notions, we introduce the context-sensitive dependency graph.

**Definition 58 (Context-Sensitive Dependency Graph [GL10a])** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The Context-Sensitive Dependency Graph (CSDG) for  $\mathcal{R}$  and  $\mu$  is  $DG(\mathcal{R}, \mu) = G(DP(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\#)$ .*

In termination proofs, we are concerned with the *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [HM05]. A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle that is not contained in any other cycle. The following result formalizes the use of SCCs for dealing with CS problems.

**Theorem 59 (SCC Processor [GL10a])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. Then, the processor  $\text{Proc}_{SCC}$  given by*

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

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

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

### 6.2.2 Estimating the Context-Sensitive Graph

Unfortunately, the context-sensitive (CS) graph in Definition 57 is not computable, since for two pairs  $u \rightarrow v$  and  $u' \rightarrow v'$ , it is undecidable whether they form a  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain. Then, for automation, an *estimated* CS graph is constructed. In the following, given a CS-TRS  $(\mathcal{P}, \mu)$  where  $\mathcal{P} = (\mathcal{G}, P)$ ,

we let  $\mathcal{P}_\mathcal{X}$  be the pairs  $u \rightarrow v \in \mathcal{P}$  such that  $v \in \mathcal{Var}(u) \setminus \mathcal{Var}^\mu(u)$ ; and  $\mathcal{P}_\mathcal{G} = \mathcal{P} \setminus \mathcal{P}_\mathcal{X}$ . When considering pairs  $u \rightarrow v \in \mathcal{P}_\mathcal{G}$ , there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$  if  $\theta(v) \hookrightarrow_{\mathcal{R}, \mu}^* \theta(u')$  for some substitution  $\theta$ . When considering collapsing pairs  $u \rightarrow v \in \mathcal{P}_\mathcal{X}$ , we know that such pairs can only be followed by a pair  $u' \rightarrow v' \in \mathcal{P}$  such that  $\theta(t) \hookrightarrow_{\mathcal{R}, \mu}^* \theta(u')$  for some  $s \rightarrow t \in \mathcal{S}_\sharp$  and substitution  $\theta$ . If  $\mathcal{S}_\sharp = \emptyset$ , then there is no outgoing arc from any  $u \rightarrow v \in \mathcal{P}_\mathcal{X}$ . Following [GTSK05], we define  $\text{TCAP}_{\mathcal{R}}^\mu$  [AGL10, Subsection 8.2]. The idea is to obtain the maximal prefix context  $C[\square]$  of  $s$  (i.e.,  $s = C[s_1, \dots, s_k]$  for some terms  $s_1, \dots, s_k$ ) that we know (without any ‘look-ahead’ for applicable rules) cannot be changed by any reduction starting from  $s$ . Furthermore, the above terms  $s_1, \dots, s_k$  must be rooted by defined symbols. Now, we replace those subterms  $s_i$  that are at  $\mu$ -replacing positions (i.e.,  $s_i = s|_{p_i}$  for some  $p_i \in \mathcal{Pos}^\mu(s)$ ) by fresh variables  $x$ , and we leave the non- $\mu$ -replacing ones untouched.

**Definition 60** [AGL10] *Given a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , we let  $\text{TCAP}_{\mathcal{R}}^\mu$  be as follows:*

$$\begin{aligned} \text{TCAP}_{\mathcal{R}}^\mu(x) &= y \text{ if } x \text{ is a variable, and} \\ \text{TCAP}_{\mathcal{R}}^\mu(f(t_1, \dots, t_k)) &= \begin{cases} f([t_1]_1^f, \dots, [t_k]_k^f) & \text{if } f([t_1]_1^f, \dots, [t_k]_k^f) \text{ does not unify} \\ & \text{with } \ell \text{ for any } \ell \rightarrow r \text{ in } \mathcal{R} \\ y & \text{otherwise} \end{cases} \end{aligned}$$

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

**Definition 61 (Estimated CS Graph [GL10a])** *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  $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ ) has  $\mathcal{P}$  as the set of nodes and arcs that connect them as follows:*

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

We have the following.

**Theorem 62 (Approximation of the CS Graph [GL10a])** *Let  $\mathcal{R}$ ,  $\mathcal{P}$  and  $\mathcal{S}$  be TRSs and  $\mu \in M_{\text{RUPUS}}$ . The estimated CS graph  $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  contains the CS graph  $\text{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ .*

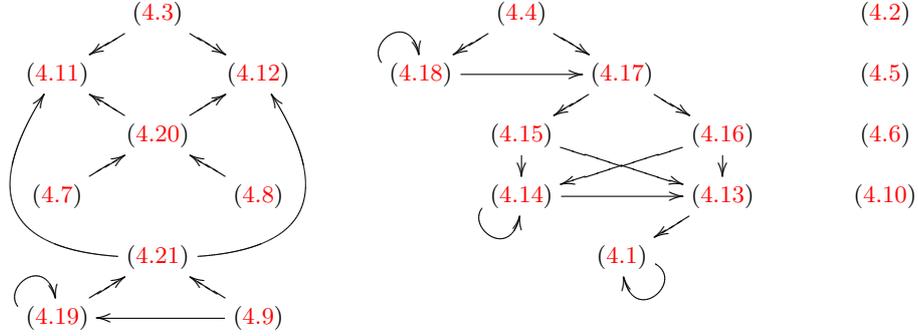


Figure 6.1: Estimated CS Dependency Graph for  $\mathcal{R}$  in Example 10

**Theorem 63 (SCC Processor using  $\text{TCAP}_{\mathcal{R}}^{\mu}$  [GL10a])** Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. The CS processor  $\text{Proc}_{\text{SCC}}$  given by

$$\text{Proc}_{\text{SCC}}(\tau) = \{(\mathcal{Q}, \mathcal{R}, \mathcal{S}_{\mathcal{Q}}, \mu) \mid \mathcal{Q} \text{ is an SCC of } \text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\}$$

where

- $\mathcal{S}_{\mathcal{Q}} = \emptyset$  if  $\mathcal{Q}_{\mathcal{X}} = \emptyset$ .
- $\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow t \mid s \rightarrow t \in \mathcal{S}_{\sharp}, \text{TCAP}_{\mathcal{R}}^{\mu}(t) \text{ and } u' \text{ unify for some } u' \rightarrow v' \in \mathcal{Q}\}$  if  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ .

is sound and complete.

The estimated CS graph of pairs is used instead of the CS graph of pairs in practice.

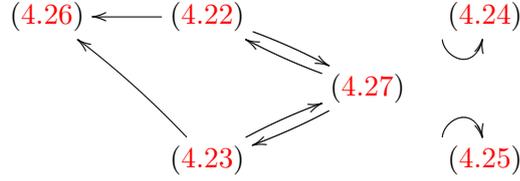
#### Example 64

In Figure 6.1, we show the estimated CS dependency graph for the CS-TRS  $(\mathcal{R}, \mu)$  in Example 52 using  $\text{TCAP}_{\mathcal{R}}^{\mu}$ . For the CS problem

$$\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$$

we have

$$\text{Proc}_{\text{SCC}}(\tau) = \{(\{(4.1)\}, \mathcal{R}, \emptyset, \mu^{\sharp}), (\{(4.14)\}, \mathcal{R}, \emptyset, \mu^{\sharp}), (\{(4.18)\}, \mathcal{R}, \emptyset, \mu^{\sharp}), (\{(4.19)\}, \mathcal{R}, \emptyset, \mu^{\sharp})\}$$

Figure 6.2: Estimated CS Dependency Graph for  $\mathcal{R}$  in Example 9**Example 65**

For Example 9, we obtain the following CS problem

$$\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\#)$$

where pairs in  $\text{DP}(\mathcal{R}, \mu)$  are given in Example 32 and  $\text{unh}(\mathcal{R}, \mu)$  is given in Example 35. The estimated CS dependency graph using  $\text{TCAP}_{\mathcal{R}}^\mu$  is depicted in Figure 6.2. The application of the SCC processor yields:

$$\begin{aligned} \text{Proc}_{\text{SCC}}(\tau) = & \{(\{(4.24)\}, \mathcal{R}, \emptyset, \mu^\#), \\ & (\{(4.22), (4.23), (4.27)\}, \mathcal{R}, \text{unh}(\mathcal{R}, \mu) \setminus \{(4.31)\}, \mu^\#), \\ & (\{(4.25)\}, \mathcal{R}, \emptyset, \mu^\#)\} \end{aligned}$$

**6.2.3 Historical Development of the CSDG**

The first notion of CSDG was first introduced in [AGL06, Subsection 4.1]. A procedure for *estimating* the graph was also defined there. We generalized the use of REN and CAP in [AG00] to CSR.

**Definition 66** Given a set  $\Delta$  of ‘defined’ symbols, we let  $\text{CAP}_{\Delta}^\mu$  be as follows:

$$\begin{aligned} \text{CAP}_{\Delta}^\mu(x) &= x \text{ if } x \text{ is a variable} \\ \text{CAP}_{\Delta}^\mu(f(t_1, \dots, t_k)) &= \begin{cases} y & \text{if } f \in \Delta \\ f([t_1]_1^f, \dots, [t_k]_1^f) & \text{otherwise} \end{cases} \end{aligned}$$

where  $y$  is intended to be a fresh variable that has not yet been used, and where given a term  $s$ ,  $[s]_i^f = \text{CAP}_{\Delta}^\mu(s)$ , if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ .

The definition of  $\text{REN}^\mu$  is the same as the one given in Definition 26.

**Example 67**

Consider the well-known example by Toyama [Toy87]:

$$f(a, b, x) \rightarrow f(x, x, x)$$

$$\begin{aligned} c &\rightarrow a \\ c &\rightarrow b \end{aligned}$$

together with  $\mu(f) = \{3\}$  and  $\mu(a) = \mu(b) = \mu(c) = \emptyset$ . The only CSDP for this system is:

$$F(a, b, x) \rightarrow F(x, x, x)$$

By using the over-approximation above, we obtain that  $\text{REN}^\mu(\text{CAP}_D^\mu(F(x, x, x))) = F(x, x, y)$  does not unify with  $F(a, b, z)$ , and thus there is no SCC.

---

In [AGL10, Subsection 8.2], we adapted Giesl et al.'s TCAP to CSR to obtain our latest over-approximation of the CAP function,  $\text{TCAP}_R^\mu$  (Definition 60).

The estimated CS graph of pairs has also evolved over time and has been improved thanks to the notions of hidden symbols, hidden terms, and hiding context.

- Our first definition of estimated CSDG in [AGL06, Section 4] treated the collapsing pairs as normal pairs. Since the right-hand sides of the collapsing pairs are variables, there was an arc from each collapsing pair to any other pair in the graph.

**Definition 68 (Estimated CSDG [AGL06])** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The estimated CSDG consists of the set  $\text{DP}(\mathcal{R}, \mu)$  of CSDPs together with arcs that connect them as follows:*

1. *There is an arc from a CSDP  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  to a CSDP  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if  $\text{REN}^\mu(\text{CAP}^\mu(v))$  and  $u'$  unify.*
2. *There is an arc from a CSDP  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to each CSDP  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$ .*

For the TRS  $\mathcal{R}$  in Example 52, the estimated CSDG following [AGL06, Section 4] is presented in Figure 6.3.

- In [AGL07], we refined the approximation. Thanks to considering connections from collapsing CSDPs to pairs whose left-hand side is rooted by hidden symbols, the number of considered arcs decreased dramatically.

**Definition 69 (Estimated CSDG [AGL07])** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The CSDG consists of the set  $\text{DP}(\mathcal{R}, \mu)$  of CSDPs together with arcs which connect them as follows:*

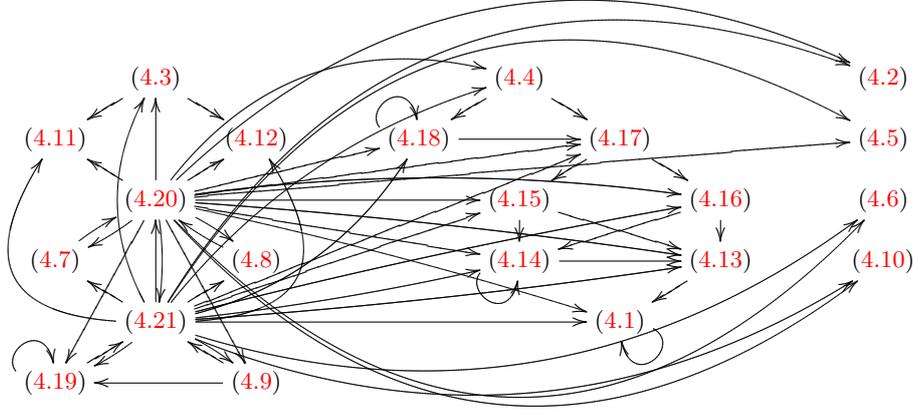


Figure 6.3: Estimated CS Graph of Pairs from Example 52 following [AGL06]

1. There is an arc from a CSDP  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  to a CSDP  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if  $\text{REN}^{\mu}(\text{CAP}^{\mu}(v))$  and  $u'$  unify.
2. There is an arc from a CSDP  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to a CSDP  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if  $\text{root}(u')^{\sharp} \in \mathcal{H}(\mathcal{R}, \mu)$ .

According to Definition 69, the estimated CSDG for the TRS  $\mathcal{R}$  in Example 52 coincides with the one depicted in Figure 6.1.

- In [AGL10, Subsection 8.1], the previous notion was improved by considering hidden terms instead of hidden symbols. Furthermore, we have adapted the notion to the CSDP framework.

**Definition 70 (Estimated CS Graph of Pairs [AGL10])** Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . The estimated CS graph associated to  $\mathcal{R}$  and  $\mathcal{P}$  (denoted  $\text{EG}(\mathcal{P}, \mathcal{R}, \mu)$ ) has  $\mathcal{P}$  as the set of nodes and the arcs that connect them as follows:

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

According to Definition 70, the estimated CSDG for the TRS  $\mathcal{R}$  in Example 52 is also depicted in Figure 6.1.

- The transformation of all collapsing pairs into noncollapsing ones as in [AEF<sup>+</sup>08, Subsection 4.1] leads to a larger graph, both in number

of nodes and in number of arcs. Also, the number of unifications that are attempted in order to build the estimated graph is greater. Figure 6.4 represents the estimated CSDG for  $\mathcal{R}$  in Example 47 according to [AEF<sup>+</sup>08].

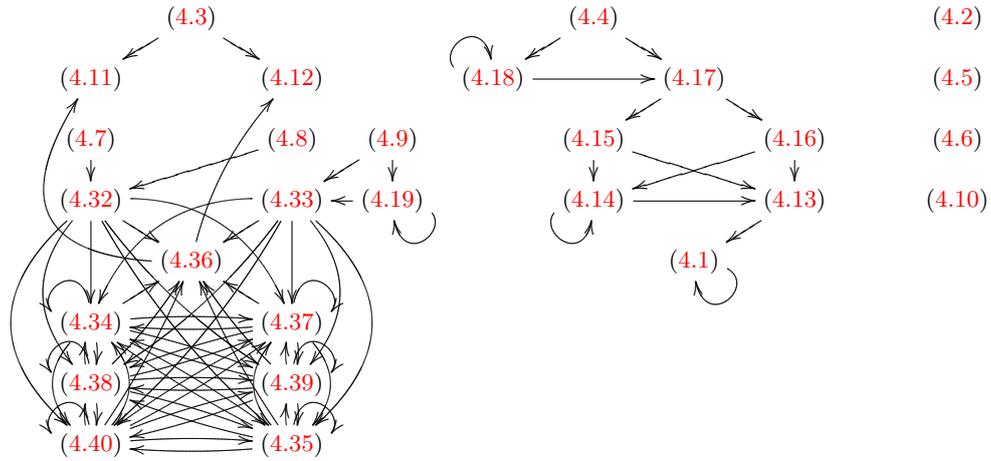


Figure 6.4: Estimated CS Graph of Pairs from Example 52 following [AEF<sup>+</sup>08]

The notion of estimated CS graph in Definition 61 was published in [GL10a].

### 6.3 Basic Processors

We have to define some trivial processors to directly establish that a CS problem is finite or infinite. The following processor helps us to check whether or not a CS problem is *infinite* by looking for embedded recursions.

**Theorem 71 (Non- $\mu$ -Termination Processor [AGL10])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. Then, the processor  $\text{Proc}_{\text{Inf}}$  given by*

$$\text{Proc}_{\text{Inf}}(\tau) = \begin{cases} \text{no} & \text{if } v = \theta(u) \\ & \text{for some } u \rightarrow v \in \mathcal{P}_G \text{ and substitution } \theta; \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

*is sound and complete.*

**Example 72**

Consider the following TRS  $\mathcal{R}$  in [AEF+08]:

$$\begin{array}{ll}
0 > y & \rightarrow \text{false} & \text{p}(0) & \rightarrow 0 \\
\text{s}(x) > 0 & \rightarrow \text{true} & \text{p}(\text{s}(x)) & \rightarrow x \\
\text{s}(x) > \text{s}(y) & \rightarrow x > y & x - y & \rightarrow \text{if}(y > 0, \text{p}(x) - \text{p}(y), x) \\
\text{if}(\text{true}, x, y) & \rightarrow x & 0 \div \text{s}(y) & \rightarrow 0 \\
\text{if}(\text{false}, x, y) & \rightarrow y & \text{s}(x) \div \text{s}(y) & \rightarrow \text{s}((x - y) \div \text{s}(y))
\end{array}$$

together with  $\mu(\text{if}) = \{1, 2\}$  and  $\mu(\text{f}) = \{1, \dots, \text{ar}(\text{f})\}$  for all other symbols  $\text{f}$ . The set of CSDPs  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\text{s}(x) >^\# \text{s}(y) \rightarrow x >^\# y \quad (6.1)$$

$$x -^\# y \rightarrow y >^\# 0 \quad (6.2)$$

$$x -^\# y \rightarrow \text{IF}(y > 0, \text{p}(x) - \text{p}(y), x) \quad (6.3)$$

$$\text{s}(x) \div^\# \text{s}(y) \rightarrow (x - y) \div^\# \text{s}(y) \quad (6.4)$$

$$\text{s}(x) \div^\# \text{s}(y) \rightarrow x -^\# y \quad (6.5)$$

$$\text{IF}(\text{false}, x, y) \rightarrow y \quad (6.6)$$

$$x -^\# y \rightarrow \text{p}(x) -^\# \text{p}(y) \quad (6.7)$$

Starting from  $\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\#)$ , the application of the processor  $\text{Proc}_{\text{Inf}}$  to  $\tau$  yields “no” because for pair (6.7), we have that  $\theta(x' -^\# y') = \text{p}(x) -^\# \text{p}(y)$  using the substitution  $\theta$  such that  $\theta(x') = \text{p}(x)$  and  $\theta(y') = \text{p}(y)$ .

The following processor exploits an old observation about collapsing CSDPs in [AGL06]. The idea is to identify a subclass of collapsing pairs which do not generate infinite chains. Thus, the notion of  $\text{DP}_{\mathcal{X}}^1$  was defined:

$$\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu) = \{f^\#(u_1, \dots, u_k) \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \mid \exists i, 1 \leq i \leq k, i \notin \mu(f^\#), x \in \text{Var}(u_i)\}$$

When considering CS problems, we generalize this notion to deal with arbitrary pairs. We say that  $f(u_1, \dots, u_k) \rightarrow x \in \mathcal{P}_{\mathcal{X}}^1$  if  $f(u_1, \dots, u_k) \rightarrow x \in \mathcal{P}_{\mathcal{X}}$  and  $x \in \text{Var}(u_i)$  for some  $i \notin \mu(f)$ . The following CS processor allow us to conclude that a CS problem is finite if all its pairs are from  $\mathcal{P}_{\mathcal{X}}^1$ .

**Theorem 73 (Basic CS Processor for Collapsing Pairs [GL10a])** *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  $\mathcal{S}_{\#}$  are noncollapsing, i.e., for all  $s \rightarrow t \in \mathcal{S}_{\#}$ ,  $t \notin \mathcal{X}$  (2)  $\{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_{\#}\} \cap \mathcal{D} = \emptyset$  and (3) for all  $s \rightarrow t \in \mathcal{S}_{\#}$ , we have that  $s = \text{f}(s_1, \dots, s_k)$  and  $t = \text{g}(s_1, \dots, s_k)$  for some  $k \in \mathbb{N}$ , function symbols  $\text{f}, \text{g} \in \mathcal{H}$ , and terms  $s_1, \dots, s_k$ . Then, the processor  $\text{Proc}_{\text{Fin}}$  given by*

$$\text{Proc}_{\text{Fin}}(\tau) = \begin{cases} \emptyset & \text{if } \mathcal{P} = \mathcal{P}_{\mathcal{X}}^1 \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Condition (3) directly emanate from [AGL06, Proposition 3], where the symbol  $\mathbf{g}$  corresponds with the marked version of  $\mathbf{f}$ .

### Example 74

Consider the following TRS  $\mathcal{R}$  [Luc98, Example 15]:

$$\begin{aligned}
0 + x &\rightarrow x \\
\mathbf{s}(x) + y &\rightarrow \mathbf{s}(x + y) \\
\mathbf{false} \wedge y &\rightarrow \mathbf{false} \\
\mathbf{true} \wedge x &\rightarrow x \\
\mathbf{first}(0, x) &\rightarrow [] \\
\mathbf{first}(\mathbf{s}(x), y : z) &\rightarrow y : \mathbf{first}(x, z) \\
\mathbf{from}(x) &\rightarrow x : \mathbf{from}(\mathbf{s}(x)) \\
\mathbf{if}(\mathbf{false}, x, y) &\rightarrow y \\
\mathbf{if}(\mathbf{true}, x, y) &\rightarrow x
\end{aligned}$$

together with  $\mu(\mathbf{first}) = \{1, 2\}$ ,  $\mu(+ ) = \mu(\wedge ) = \mu(\mathbf{if}) = \{1\}$ , and  $\mu(\mathbf{from}) = \mu(0) = \mu(:) = \mu(\mathbf{false}) = \mu([]) = \mu(\mathbf{s}) = \mu(\mathbf{true}) = \emptyset$ . We have that  $\text{DP}(\mathcal{R}, \mu)$  consists of the following rules:

$$\begin{aligned}
0 +^\sharp x &\rightarrow x \\
\mathbf{true} \wedge^\sharp x &\rightarrow x \\
\mathbf{IF}(\mathbf{false}, x, y) &\rightarrow y \\
\mathbf{IF}(\mathbf{true}, x, y) &\rightarrow x
\end{aligned}$$

Note that all CSDPs are collapsing; furthermore,  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_\chi^1(\mathcal{R}, \mu)$ . The unhiding TRS  $\text{unh}(\mathcal{R}, \mu)$  is:

$$\begin{aligned}
x + y &\rightarrow x +^\sharp y \\
x + y &\rightarrow x \\
\mathbf{first}(x, z) &\rightarrow \mathbf{FIRST}(x, z) \\
\mathbf{first}(x, z) &\rightarrow x \\
\mathbf{first}(x, z) &\rightarrow z \\
\mathbf{from}(\mathbf{s}(x)) &\rightarrow \mathbf{FROM}(\mathbf{s}(x))
\end{aligned}$$

Let  $\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ . Since  $\text{Proc}_{\text{Fin}}(\tau) = \emptyset$ , we conclude that  $\tau$  is finite.

Processor  $\text{Proc}_{\text{Fin}}$  is also used later on in Example 79.

## 6.4 Treating Collapsing Pairs

As we have discussed in Section 4.4, the idea of having only ‘standard’ pairs instead of two kind of pairs was considered in the earlier stages of the development of this thesis. We can integrate the transformation in Definition 46 as a processor in our CSDP framework. The idea is the same as in the definition of the CSDPs in [AEF<sup>+</sup>08]: the inclusion of a fresh symbol  $U$  that encapsulates the unhiding TRS as pairs.

**Theorem 75 (Collapsing Pair Transformation Processor [GL10a])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem where  $\mathcal{P} = (\mathcal{G}, P)$  and, let  $P_U$  be given by the following rules:*

- $u \rightarrow U(x)$  for every  $u \rightarrow x \in \mathcal{P}_\mathcal{X}$ ,
- $U(s) \rightarrow U(t)$  for every  $s \rightarrow t \in \mathcal{S}_{\triangleright\mu}$ , and
- $U(s) \rightarrow t$  for every  $s \rightarrow t \in \mathcal{S}_\sharp$ .

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

### Example 76

Continuing with Example 65, for the CS problem

$$\tau = \{(\{(4.22), (4.23), (4.27)\}, \mathcal{R}, \text{unh}(\mathcal{R}, \mu) \setminus \{(4.31)\}, \mu^\sharp)\}$$

we obtain

$$\{(\{(6.8), (6.9), (4.27), (6.10), (6.11), (6.12)\}, \mathcal{R}, \emptyset, \mu^\sharp)\}$$

where the new transformed pairs are:

$$\text{IF}(\text{false}, x, y) \rightarrow U(y) \tag{6.8}$$

$$\text{IF}(\text{true}, x, y) \rightarrow U(x) \tag{6.9}$$

$$U(\text{s}(x)) \rightarrow U(x) \tag{6.10}$$

$$U(x \div y) \rightarrow U(x) \tag{6.11}$$

$$U((x - y) \div \text{s}(y)) \rightarrow (x - y) \div^\sharp \text{s}(y) \tag{6.12}$$

with  $\mu$  extended by  $\mu(U) = \emptyset$ .

## 6.5 Historical Development of the Collapsing Pair Transformation

As explained in Section 4.4, transforming collapsing pairs is as old as the definition of CSDP. In the first preliminary version of CSDPs in Definition 39, the attempt to avoid collapsing pairs forces us to include a pair of the form  $\text{MUSUBTERM}(f(x_1, \dots, x_i, \dots, x_k)) \rightarrow \text{MUSUBTERM}(x_i)$  for each function symbol  $f$  in the signature and position  $i \in \mu(f)$  and a pair  $\text{MUSUBTERM}(f(x_1, \dots, x_k)) \rightarrow F(x_1, \dots, x_k)$  for each defined function symbol  $f$ . The transformation used in [AEF<sup>+</sup>08] is similar to the one defined in this thesis, but with the difference that it is applied at the very beginning, in the definition of CSDP, using the notion of hidden term and hiding context (see Definition 46).

Since the most basic notion when *modeling* the termination behavior of CSR is that of collapsing pair and unhiding TRS, the transformation should be considered as an ingredient for handling collapsing pairs in proofs of termination (as implemented by the processor  $\text{Proc}_{eColl}$  above).

## 6.6 Reduction Triple Processor

In Section 6.1 we have introduced reduction pairs. Reduction pairs help us to discard pairs from DP problems. A reduction pair processor tries to find a valid reduction pair where some pairs in  $\mathcal{P}$  are *strictly oriented*; then, we can ensure that these pairs are not involved in any infinite chain.

In the CSDP framework, since we have three different TRSs, it is normal to have three different relations instead of two. Therefore, we can use  $\mu$ -reduction triples:

**Definition 77 ( $\mu$ -Reduction Triple [GL10a])** *Let  $\mathcal{F}$  be a signature and  $\mu \in M_{\mathcal{F}}$ . A  $\mu$ -reduction triple  $(\succsim, \sqsupset, \succeq)$  consists of a  $\mu$ -reduction pair  $(\succsim, \sqsupset)$  and a stable relation on terms  $\succeq$ , which is compatible with  $\succsim$  or  $\sqsupset$ , i.e.,  $\succeq \circ \succsim \subseteq \succsim$  or  $\sqsupset \circ \succeq \subseteq \sqsupset$ . We say that  $(\succsim, \sqsupset, \succeq)$  is  $\mu$ -monotonic if  $\sqsupset$  is  $\mu$ -monotonic.*

The new component  $\succeq$  is intended to deal with the rules in  $\mathcal{S}$ . Note that only stability is required for  $\succeq$ : since all the steps in  $\mathcal{S}$  are at the root position, no monotonicity requirements are necessary for this relation.

**Theorem 78 ( $\mu$ -Reduction Triple Processor [GL10a])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. Let  $(\succsim, \sqsupset, \succeq)$  be a  $\mu$ -reduction triple such that*

1.  $\mathcal{P} \subseteq \succsim \cup \sqsupset$ ,  $\mathcal{R} \subseteq \succsim$ , and

2. whenever  $\mathcal{P}_X \neq \emptyset$  we have that  $\mathcal{S} \subseteq \succeq \cup \sqsupset \cup \succeq$ .

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

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

is sound and complete.

These  $\mu$ -reduction triples can be used in combination with *argument filterings*, which discard subexpressions from constraints  $s \succeq t$ ,  $s \sqsupset t$  or  $s \succeq t$  in such a way that  $\pi(s) \succeq \pi(t)$  (resp.  $\pi(s) \sqsupset \pi(t)$  or  $\pi(s) \succeq \pi(t)$ ) is often simpler to prove [AG00, GTSKF06].

An argument filtering  $\pi$  for a signature  $\mathcal{F}$  is a mapping that assigns to every  $k$ -ary function symbol  $f \in \mathcal{F}$  an argument position  $i \in \{1, \dots, k\}$  or a (possibly empty) list  $[i_1, \dots, i_m]$  of argument positions with  $1 \leq i_1 < \dots < i_m \leq k$  [KNT99].

### Example 79

Continuing Example 65, for the CS problem

$$\tau_0 = \{(\mathcal{P}_0, \mathcal{R}, \mathcal{S}_0, \mu^\sharp)\}$$

where  $\mathcal{P}_0 = \{(4.22), (4.23), (4.27)\}$  and  $\mathcal{S}_0 = \text{unh}(\mathcal{R}, \mu) \setminus \{(4.31)\}$ , we iterate on  $\text{Proc}_{RT}$  as follows. With the following automatically generated polynomial interpretation:

$$\begin{array}{ll} [\div](x, y) = x + 1 & [\geq](x, y) = x \\ [\text{if}](x, y, z) = x + y + z & [-](x, y) = 0 \\ [0] = 0 & [\text{s}](x) = x \\ [\text{false}] = 0 & [\text{true}] = 0 \\ [\div^\sharp](x, y) = x + 1 & [\text{IF}](x, y, z) = x + y + z \end{array}$$

we obtain  $\text{Proc}_{RT}(\tau_0) = \{\tau_1\}$ , where:

$$\tau_1 = \{(\mathcal{P}_0, \mathcal{R}, \mathcal{S}_1, \mu^\sharp)\}.$$

where  $\mathcal{S}_1 = \mathcal{S}_0 \setminus \{(4.28)\}$ . For  $\tau_1$ , we obtain a new automatically generated polynomial interpretation:

$$\begin{array}{ll} [\div](x, y) = x + y + 1 & [\geq](x, y) = x + y + 1 \\ [\text{if}](x, y, z) = y + z & [-](x, y) = x \\ [0] = 0 & [\text{s}](x) = x + 1 \\ [\text{false}] = 0 & [\text{true}] = 1 \\ [\div^\sharp](x, y) = x + y + 1 & [\text{IF}](x, y, z) = y + z \end{array}$$

that allows us to apply  $\text{Proc}_{RT}$  again to obtain a new CS problem  $\tau_2$  where the rule (4.29) is removed from  $\mathcal{S}_1$ , thus leaving  $\mathcal{S}_{2,\triangleright_\mu}$  empty:

$$\tau_2 = \{(\mathcal{P}_0, \mathcal{R}, \mathcal{S}_2, \mu^\sharp)\}.$$

where  $\mathcal{S}_2 = \mathcal{S}_1 \setminus \{(4.29)\}$ . We can apply  $\text{Proc}_{RT}$  again over  $\tau_2$  to remove the pair (4.27) with the following polynomial interpretation:

$$\begin{array}{ll} [\div](x, y) = x + y + 1 & [\geq](x, y) = 0 \\ [\text{if}](x, y, z) = x + y + z + 1 & [-](x, y) = 1 \\ [0] = 1 & [\text{s}](x) = 1 \\ [\text{false}] = 0 & [\text{true}] = 0 \\ [\div^\sharp](x, y) = x + y + 1 & [\text{IF}](x, y, z) = x + y + z \end{array}$$

then we obtain the CS problem:

$$\tau_3 = \{(\mathcal{P}_3, \mathcal{R}, \mathcal{S}_2, \mu^\sharp)\}$$

where  $\mathcal{P}_3 = \mathcal{P}_0 \setminus \{(4.24)\} = \mathcal{P}_{3,\mathcal{X}}^1$ . Now, we can conclude finiteness of  $\tau_3$  by using  $\text{Proc}_{Fin}$ . Hence, finiteness of  $\tau_0$  is finally proved.

### 6.6.1 Historical Development of $\mu$ -Reduction Triples

In [AGL06, AGL10], when a collapsing pair  $u \rightarrow x$  occurs in a chain, we have to *look inside* the instantiated right-hand side  $\sigma(x)$  for a  $\mu$ -replacing subterm that, after being marked,  $\mu$ -rewrites to the (instantiated) left-hand side of another CSDP. For this reason, the quasi-orderings  $\succsim$  of reduction pairs ( $\succsim, \sqsupset$ ) which are used in [AGL06, AGL10] are required to have the  $\mu$ -subterm property, i.e.  $\triangleright_\mu \subseteq \succsim$ . This is equivalent to imposing  $f(x_1, \dots, x_k) \succsim x_i$  for all projection rules  $f(x_1, \dots, x_k) \rightarrow x_i$  with  $f \in \mathcal{F}$  and  $i \in \mu(f)$ . This is similar for markings: in [AGL06] we have to ensure that  $f(x_1, \dots, x_k) \succsim f^\sharp(x_1, \dots, x_k)$  for all defined symbols  $f$  in the signature. In [AGL10], thanks to the notion of hidden term, we relaxed the last condition: we require  $t \succsim t^\sharp$  for all (narrowable) *hidden terms*  $t$ . In [AEF<sup>+</sup>08], thanks to the notion of hiding context, we only require that  $\succsim$  be compatible with the projections  $f(x_1, \dots, x_k) \rightarrow x_i$  for those symbols  $f$  and positions  $i$  such that  $f$  *hides position*  $i$ . However, this information is implicitly encoded as (new) pairs  $U(f(x_1, \dots, x_k)) \rightarrow U(x_i)$  in the set  $\mathcal{P}$ . The strict component  $\sqsupset$  of the reduction pair ( $\succsim, \sqsupset$ ) is now used with these new pairs.

In the current approach, since the rules in  $\mathcal{S}$  are not considered to be ordinary pairs (in the sense of [AEF<sup>+</sup>08, AGL10]), we can relax the conditions

imposed on the orderings that deal 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 that is compatible with these rules (no transitivity, reflexivity, well-foundedness or  $\mu$ -monotonicity is required). Another advantage is that we can now *remove rules from  $\mathcal{S}$* .

## 6.7 Reduction Triple Processor with Usable Rules

In order to use  $\text{Proc}_{RT}$ , we require the rules in the TRS  $\mathcal{R}$  of the CS problem  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  to be included in  $\succsim$ , i.e.,  $\mathcal{R} \subseteq \succsim$  must hold. However, it would be desirable to consider only the rules that are really necessary to capture all possible infinite sequences instead of *all* rules  $\mathcal{R}$  in the CS problem. *Usable rules* [AG00, HM04, TGSK04] provide a sound estimation of this ‘minimal’ set.

Usable rules were introduced by Arts and Giesl in [AG00] in connection with innermost termination. Hirokawa and Middeldorp [HM04] and (independently) Thiemann et al. [TGSK04] showed how to use them to prove termination of rewriting.

In order to adapt the notion of usable rules to CSR, we follow the classical approaches in [GTSKF06, HM07], which are based on the notion of *dependency* among function symbols. Let  $\text{rules}_{\mathcal{R}}(\mathbf{f}) = \{\ell \rightarrow r \in \mathcal{R} \mid \text{root}(\ell) = \mathbf{f}\}$ . The set of  $\mu$ -replacing symbols in a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is denoted by  $\text{Fun}^{\mu}(t) = \{\mathbf{f} \mid \exists p \in \text{Pos}^{\mu}(t), \mathbf{f} = \text{root}(t|_p)\}$ . The simplest adaptation of this notion is the following.

**Definition 80 (Basic  $\mu$ -Dependency [GLU08])** *Given a TRS  $(\mathcal{F}, R)$  and  $\mu \in M_{\mathcal{F}}$ , we say that  $\mathbf{f} \in \mathcal{F}$  has a basic  $\mu$ -dependency on  $\mathbf{h} \in \mathcal{F}$  (written  $\mathbf{f} \blacktriangleright_{\mathcal{R}, \mu} \mathbf{h}$ ) if  $\mathbf{f} = \mathbf{h}$  or there is a function symbol  $\mathbf{g}$  with  $\mathbf{g} \blacktriangleright_{\mathcal{R}, \mu} \mathbf{h}$  and a rule  $\ell \rightarrow r \in \text{rules}_{\mathcal{R}}(\mathbf{f})$  with  $\mathbf{g} \in \text{Fun}^{\mu}(r)$ .*

The corresponding notion of *basic CS usable rule* is the following.

**Definition 81 (Basic CS Usable Rules [GL10b])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. The set  $\mathcal{U}^{\blacktriangleright}(\tau)$  of basic context-sensitive usable rules of  $\tau$  is*

$$\mathcal{U}^{\blacktriangleright}(\tau) = \bigcup_{u \rightarrow v \in \mathcal{P}, \mathbf{f} \in \text{Fun}^{\mu}(v), \mathbf{f} \blacktriangleright_{\mathcal{R}, \mu} \mathbf{g}} \text{rules}_{\mathcal{R}}(\mathbf{g})$$

However, Definition 81 does *not* lead to a correct approach for proving termination of CSR.

**Example 82**

Consider the TRS  $\mathcal{R}$  [AL07]:

$$\begin{aligned} f(c(x), x) &\rightarrow f(x, x) \\ \mathbf{b} &\rightarrow c(\mathbf{b}) \end{aligned}$$

together with  $\mu(f) = \{1, 2\}$  and  $\mu(c) = \emptyset$ . We have the following set of CSDPs  $\text{DP}(\mathcal{R}, \mu)$ :

$$F(c(x), x) \rightarrow F(x, x)$$

The unhiding TRS  $\text{unh}(\mathcal{R}, \mu)$  is:

$$\mathbf{b} \rightarrow \mathbf{B}$$

Since there is no collapsing pair in  $\text{DP}(\mathcal{R}, \mu)$ , after applying  $\text{Proc}_{SCC}(\tau_0)$ , where  $\tau_0 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\#)$ , we obtain

$$\tau_1 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\#)$$

According to Definition 81, we have no basic usable rules for  $\tau$  because  $F(x, x)$  contains no symbol in  $\mathcal{F}$ . We could wrongly conclude finiteness of the CS problem and, hence,  $\mu$ -termination of  $(\mathcal{R}, \mu)$ , but we have the infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\#)$ -chain where  $\mathbf{b} \rightarrow c(\mathbf{b})$  is *used*:

$$\underline{F(c(\mathbf{b}), \mathbf{b})} \hookrightarrow_{\mathcal{P}, \mu^\#} \underline{F(\mathbf{b}, \mathbf{b})} \hookrightarrow_{\mathcal{R}, \mu^\#} \underline{F(c(\mathbf{b}), \mathbf{b})} \hookrightarrow_{\mathcal{P}, \mu^\#} \dots$$

Although basic usable rules are not correct for any kind of CS problem (as we show below), they can be used in presence of *strong conservativity*.

**Definition 83 (Strong Conservativity [GLU08])** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . A rule  $\ell \rightarrow r$  is strongly  $\mu$ -conservative if it is  $\mu$ -conservative and  $\text{Var}^\mu(\ell) \cap \text{Var}^{\neq\mu}(\ell) = \text{Var}^\mu(r) \cap \text{Var}^{\neq\mu}(r) = \emptyset$ ; and  $\mathcal{R}$  is strongly  $\mu$ -conservative if all rules in  $\mathcal{R}$  are strongly  $\mu$ -conservative.*

In order to obtain an appropriate and general definition of usable rule for CSR, we have to consider the rules of *symbols* in hidden terms (that is, the *hidden symbols*) as usable. We also extend the notion of  $\mu$ -dependency to capture the usable rules when collapsing pairs are present:

**Example 84**

Consider the following CS problem  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  where  $\mathcal{P}$  is:

$$\begin{aligned} G(a) &\rightarrow F(g(b)) \\ F(x) &\rightarrow x \end{aligned}$$

the only rule in  $\mathcal{R}$  is:

$$b \rightarrow a$$

and  $\mathcal{S}$  consists of a single rule as well:

$$g(x) \rightarrow G(x)$$

Let  $\mu$  be given by  $\mu(g) = \mu(G) = \{1\}$  and  $\mu(F) = \mu(a) = \mu(b) = \emptyset$ . Since we have a collapsing pair, the rule in  $\mathcal{R}$  is usable because it is necessary to build the following infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain:

$$\underline{G(a)} \hookrightarrow_{\mathcal{P}, \mu} \underline{F(g(b))} \hookrightarrow_{\mathcal{P}, \mu} \circ \overset{\Lambda}{\hookrightarrow}_{\mathcal{S}, \mu} \underline{G(b)} \hookrightarrow_{\mathcal{R}, \mu} \underline{G(a)} \hookrightarrow_{\mathcal{P}, \mu} \dots$$

But, even taking into account the hidden symbols and extending the notion of  $\mu$ -dependency, we do not get a correct definition of usable rule:

**Example 85**

Consider the following ( $\mu$ -conservative) non- $\mu$ -terminating CS-TRS  $\mathcal{R}$ :

$$\begin{aligned} a(x, y) &\rightarrow b(x, x) \\ d(x, e) &\rightarrow a(x, x) \\ b(x, g) &\rightarrow d(x, x) \\ g &\rightarrow e \end{aligned}$$

with  $\mu(a) = \mu(d) = \{1, 2\}$ ,  $\mu(b) = \{1\}$  and  $\mu(g) = \mu(e) = \emptyset$ . The set  $\text{DP}(\mathcal{R}, \mu)$  of CSDPs is:

$$\begin{aligned} A(x, y) &\rightarrow B(x, x) \\ D(x, e) &\rightarrow A(x, x) \\ B(x, g) &\rightarrow D(x, x) \end{aligned}$$

and, since  $\text{unh}(\mathcal{R}, \mu)$  is empty, finiteness of the following CS problem:

$$\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\sharp)$$

is equivalent to  $\mu$ -termination of  $\mathcal{R}$ . According to Definition 81, we have no basic usable rules because the right-hand sides of the DPs have no defined symbols, and we have no hidden symbol since there is no hidden term.

In order to use the usable rules instead of all the rules, we add to  $\mathcal{R}$  the following  $(\mathcal{C}_\varepsilon)$  rules:

$$\begin{aligned} c(x, y) &\rightarrow x \\ c(x, y) &\rightarrow y \end{aligned}$$

where  $c$  is a fresh binary function symbol which allows us to simulate the application of the “removed” rules. In this way, if we consider no replacement restrictions, the rule  $g \rightarrow e$  is not needed to capture the following infinite chain:

$$\underline{A(g, g)} \rightarrow_{\mathcal{P}} \underline{B(g, g)} \rightarrow_{\mathcal{P}} \underline{D(g, g)} \rightarrow_{\mathcal{R}} \underline{D(g, e)} \rightarrow_{\mathcal{P}} \underline{A(g, g)} \rightarrow_{\mathcal{P}} \dots$$

because we have the following sequence using  $\mathcal{C}_\varepsilon$ -rules instead:

$$\begin{aligned} \underline{A(c(g, e), c(g, e))} &\rightarrow_{\mathcal{P}} \underline{B(c(g, e), c(g, e))} \rightarrow_{\mathcal{C}_\varepsilon} \underline{B(c(g, e), g)} \rightarrow_{\mathcal{P}} \\ \underline{D(c(g, e), c(g, e))} &\rightarrow_{\mathcal{C}_\varepsilon} \underline{D(c(g, e), e)} \rightarrow_{\mathcal{P}} \underline{A(c(g, e), c(g, e))} \rightarrow_{\mathcal{P}} \dots \end{aligned}$$

However, if we consider the replacement restrictions now, the  $\mu$ -rewrite step

$$\underline{B(c(g, e), c(g, e))} \hookrightarrow_{\mathcal{C}_\varepsilon} \underline{B(c(g, e), g)}$$

is no longer possible. Since the infinite sequence above can be regarded as an infinite  $\mu$ -rewrite sequence:

$$\underline{A(g, g)} \hookrightarrow_{\mathcal{P}, \mu} \underline{B(g, g)} \hookrightarrow_{\mathcal{P}, \mu} \underline{D(g, g)} \hookrightarrow_{\mathcal{R}, \mu} \underline{D(g, e)} \hookrightarrow_{\mathcal{P}, \mu} \underline{A(g, g)} \hookrightarrow_{\mathcal{P}, \mu} \dots$$

In order to avoid this problem, we modify Definition 80 to take into account symbols that occur at frozen positions in the *left-hand sides* of the rules. The set of *non- $\mu$ -replacing* symbols in a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is denoted by  $\mathcal{F}un^\mu(t) = \{f \mid \exists p \in \mathcal{P}os(t) \setminus \mathcal{P}os^\mu(t), f = \text{root}(t|_p)\}$ .

**Definition 86 ( $\mu$ -Dependency [GLU08])** Given a TRS  $(\mathcal{F}, \mathcal{R})$  and  $\mu \in M_{\mathcal{F}}$ , we say that  $f \in \mathcal{F}$  has a  $\mu$ -dependency on  $h \in \mathcal{F}$ , written  $f \triangleright_{\mathcal{R}, \mu} h$ , if  $f = h$  or there is a function symbol  $g$  with  $g \triangleright_{\mathcal{R}, \mu} h$  and a rule  $\ell \rightarrow r \in \text{rules}_{\mathcal{R}}(f)$  with  $g \in \mathcal{F}un^\mu(\ell) \cup \mathcal{F}un(r)$ .

We adapt the notion of usable rules to deal with any kind of CS problem.

**Definition 87 (CS Usable Rules [GL10b])** Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. The set  $\mathcal{U}^\triangleright(\tau)$  of context-sensitive usable rules of  $\tau$  is

$$\begin{aligned} \mathcal{U}^\triangleright(\tau) = & \bigcup_{u \rightarrow v \in \mathcal{P}, f \in \mathcal{F}un^\mu(u) \cup \mathcal{F}un(v), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_{\mathcal{R}}(g) \cup \\ & \bigcup_{\ell \rightarrow r \in \mathcal{R}, f \in \mathcal{F}un^\mu(r), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_{\mathcal{R}}(g) \cup \\ & \bigcup_{s \rightarrow t \in \mathcal{S}, f \in \mathcal{F}un(s) \cup \mathcal{F}un(t), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_{\mathcal{R}}(g) \end{aligned}$$

Now, we can define a valid processor for  $\mu$ -reduction triples with usable rules. In order to formulate it, we have to consider the so-called  $\mathcal{C}_\varepsilon$ -compatibility of  $\succsim$ , i.e.,  $c(x, y) \succsim x$  and  $c(x, y) \succsim y$  holds for a fresh function symbol  $c$  [GL02a].

**Theorem 88** ( $\mu$ -Reduction Triple Processor with Usable Rules [GL10b])

Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. Let  $(\succsim, \sqsupset, \succ)$  be a  $\mu$ -reduction triple such that

1.  $\mathcal{P} \subseteq \succsim \cup \sqsupset$ ,
2. at least one of the following holds:
  - $\mathcal{U}^\blacktriangleright(\tau) \subseteq \succsim$ ,  $\mathcal{P}\mathcal{U}^\blacktriangleright(\tau)$  is strongly  $\mu$ -conservative,  $\succsim$  is  $\mathcal{C}_\varepsilon$ -compatible
  - $\mathcal{U}^\blacktriangleright(\tau) \subseteq \succsim$ ,  $\succsim$  is  $\mathcal{C}_\varepsilon$ -compatible,
  - $\mathcal{R} \subseteq \succsim$ ,
3. and, whenever  $\mathcal{P}_\mathcal{X} \neq \emptyset$ , we have that  $\mathcal{S} \subseteq \succsim \cup \sqsupset \cup \succ$ .

Let  $\mathcal{P}_\sqsupset = \{u \rightarrow v \in \mathcal{P} \mid u \sqsupset v\}$  and  $\mathcal{S}_\sqsupset = \{s \rightarrow t \in \mathcal{S} \mid s \sqsupset t\}$ . Then, the processor  $\text{Proc}_{UR}$  given by

$$\text{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_\sqsupset, \mu)\} & \text{if (1), (2) and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Example 89**

Assume that we want to compute the zip of two lists by applying the quotient of its components [Bor03]. The following rule can be used to generate the list of all natural numbers:

$$\text{from}(x) \rightarrow x : \text{from}(s(x)) \quad (6.13)$$

The evaluation of  $\text{from}(0)$  would produce the infinite list  $0 : s(0) : s(s(0)) : \dots$  of all natural numbers. The  $n$ -th component of a finite or infinite list can be retrieved by using the following function  $\text{sel}$ :

$$\text{sel}(0, x : xs) \rightarrow x \quad (6.14)$$

$$\text{sel}(s(n), x : xs) \rightarrow \text{sel}(n, xs) \quad (6.15)$$

We define the *zip with quot* ( $\text{zWquot}$ ) function by means of rules in the following way:

$$x - 0 \rightarrow 0 \quad (6.16)$$

$$\mathfrak{s}(x) - \mathfrak{s}(y) \rightarrow x - y \quad (6.17)$$

$$0 \div \mathfrak{s}(y) \rightarrow 0 \quad (6.18)$$

$$\mathfrak{s}(x) \div \mathfrak{s}(y) \rightarrow \mathfrak{s}((x - y) \div \mathfrak{s}(y)) \quad (6.19)$$

$$\mathfrak{zWquot}(xs, []) \rightarrow [] \quad (6.20)$$

$$\mathfrak{zWquot}([], xs) \rightarrow [] \quad (6.21)$$

$$\mathfrak{zWquot}(x : xs, y : ys) \rightarrow (x \div y) : \mathfrak{zWquot}(xs, ys) \quad (6.22)$$

If our strategy is eager, the evaluation of any expression containing a call to `from` starts an infinite computation. This is due to the recursive call to `from` in the second argument of the list constructor `(:)` in the right-hand side of the rule (6.13). Then, we have to evaluate the second argument of `(:)` only when needed. Consider the replacement map  $\mu$  for the signature  $\mathcal{F}$  (consisting of the function symbols above) given by:  $\mu(\cdot) = \{1\}$  and  $\mu(\mathfrak{f}) = \{1, \dots, \text{ar}(\mathfrak{f})\}$  for all  $\mathfrak{f} \in \mathcal{F} \setminus \{\cdot\}$ . Thanks to the replacement map, we avoid the following DP:

$$\text{from}(x) \rightarrow \text{FROM}(\mathfrak{s}(x))$$

which represents the harmful recursive call. We have to consider the following set of CSDPs  $\text{DP}(\mathcal{R}, \mu)$ :

$$\text{SEL}(\mathfrak{s}(n), x : xs) \rightarrow \text{SEL}(n, xs) \quad (6.23)$$

$$\text{SEL}(\mathfrak{s}(n), x : xs) \rightarrow xs \quad (6.24)$$

$$\mathfrak{s}(x) -^{\#} \mathfrak{s}(y) \rightarrow x -^{\#} y \quad (6.25)$$

$$\mathfrak{s}(x) \div^{\#} \mathfrak{s}(y) \rightarrow (x - y) \div^{\#} \mathfrak{s}(y) \quad (6.26)$$

$$\mathfrak{s}(x) \div^{\#} \mathfrak{s}(y) \rightarrow x -^{\#} y \quad (6.27)$$

$$\text{ZwQUOT}(x : xs, y : ys) \rightarrow x \div^{\#} y \quad (6.28)$$

The unhiding TRS  $\text{unh}(\mathcal{R}, \mu)$  consists of the following rules:

$$\text{from}(\mathfrak{s}(x)) \rightarrow \text{FROM}(\mathfrak{s}(x))$$

$$\mathfrak{zWquot}(xs, ys) \rightarrow \text{ZwQUOT}(xs, ys)$$

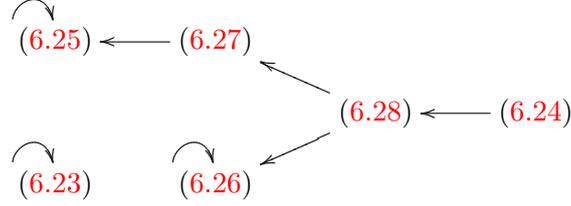
$$\mathfrak{zWquot}(x, y) \rightarrow x$$

$$\mathfrak{zWquot}(x, y) \rightarrow y$$

We can now define the CS problem

$$\tau_0 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\#})$$

The estimated CSDG  $\text{EG}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\#})$  for  $\tau_0$  is depicted in Figure 6.5. If we apply the SCC processor, we obtain:

Figure 6.5: Estimated CS Dependency Graph from  $\tau_0$  in Example 89

$$\text{Proc}_{SCC}(\tau_0) = \{\tau_1, \tau_2, \tau_3\}$$

where  $\tau_1$ ,  $\tau_2$  and  $\tau_3$  are the CS problems  $\tau_1 = (\{(6.23)\}, \mathcal{R}, \emptyset, \mu^\sharp)$ ,  $\tau_2 = (\{(6.25)\}, \mathcal{R}, \emptyset, \mu^\sharp)$  and  $\tau_3 = (\{(6.26)\}, \mathcal{R}, \emptyset, \mu^\sharp)$ .

For the CS problem  $\tau_1$  we can apply  $\text{Proc}_{UR}$ . For  $\tau_1$  we have that:

$$\begin{aligned} \mathcal{U}^\blacktriangleright(\tau_1) &= \emptyset \\ \mathcal{U}^\triangleright(\tau_1) &= \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\} \end{aligned}$$

Since (6.23) is not strongly  $\mu$ -conservative, we use  $\mathcal{U}^\triangleright(\tau_1)$  and the following polynomial interpretation<sup>1</sup>:

$$\begin{array}{ll} [\text{SEL}](x, y) = x + y & [-](x, y) = x \\ [\div](x, y) = x + y + 1 & [\text{zWquot}](x, y) = x + y + 1 \\ [ : ](x, y) = y & [ [ ] ] = 1 \\ [0] = 0 & [s](x) = x + 1 \end{array}$$

to conclude finiteness of  $\tau_1$ .

For the CS problem  $\tau_2$  we can apply  $\text{Proc}_{UR}$  as well. For  $\tau_2$  we have that:

$$\begin{aligned} \mathcal{U}^\blacktriangleright(\tau_2) &= \emptyset \\ \mathcal{U}^\triangleright(\tau_2) &= \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\} \end{aligned}$$

Since  $\{(6.25)\} \cup \mathcal{U}^\blacktriangleright(\tau_2)$  is strongly  $\mu$ -conservative, the set of usable rules is empty and we can use the following polynomial interpretation:

$$[-^\sharp](x, y) = x + y \quad [s](x) = x + 1$$

to conclude finiteness of  $\tau_2$ .

For the CS problem  $\tau_3$  we can apply  $\text{Proc}_{UR}$  again. For  $\tau_3$  we have that:

$$\begin{aligned} \mathcal{U}^\blacktriangleright(\tau_3) &= \{(6.16), (6.17)\} \\ \mathcal{U}^\triangleright(\tau_3) &= \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\} \end{aligned}$$

<sup>1</sup>The quasi-orderings  $\succsim$  induced by a polynomial interpretation are always  $\mathcal{C}_e$ -compatible.

Since  $\{(6.26)\} \cup \mathcal{U}^\blacktriangleright(\tau_3)$  is strongly  $\mu$ -conservative, we can use the following polynomial interpretation:

$$\begin{array}{ll} [\div^\sharp](x, y) = x & [-](x, y) = x \\ [0] = 0 & [s](x) = x + 1 \end{array}$$

to conclude finiteness of  $\tau_3$ .

### 6.7.1 Historical Development of $\mu$ -Reduction Triples with Usable Rules

The notion of usable rule in connection with CSR was investigated first in [AL07] to prove innermost termination of CSR. In this work, the authors defined the basic usable rules (essentially, those in Definition 81) which are valid for any kind of  $\mu$ -conservative system. However, as we discussed in [GLU08], even for this restricted setting, the results obtained for innermost termination of CSR cannot be directly adapted to termination of CSR. As Example 82 shows,  $\mathcal{R}$  is  $\mu$ -conservative and the notion of basic usable rule can be used if we want to prove innermost termination (indeed,  $\mathcal{R}$  is innermost  $\mu$ -terminating); however, the notion of basic usable rule is not valid if we want to prove termination of  $\mathcal{R}$ .

In [GLU08], we investigated how to obtain the set of usable rules for CSR. In this work, we obtained two notions of usable rules: a totally general one, which can be used with any kind of CS-TRS; and a more restricted one, which can be used with strongly  $\mu$ -conservative CS-TRSs and uses the notion of basic usable rule. The proof of soundness in this work relies on a transformation in which all infinite (minimal) rewrite sequences can be simulated by using a *restricted* set of rules. This transformation was devised by Gramlich for a completely different purpose [Gra94]. Later on, Urbain [Urb04] used it (with some modifications) to prove termination of rewriting modules. Finally, Hirokawa and Middeldorp [HM04] and (independently) Thiemann *et al.* [TGSK04] combined this idea with the idea of *usable rules* leading to an improved framework for proving termination of rewriting using DPs.

**Definition 90 (Interpretation [TGSK04, HM04])** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\Delta \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F} \cup \{\perp, c\}, \mathcal{X})$  where  $\perp$  is a fresh constant symbol and  $c$  is a fresh binary symbol. The interpretation  $\mathcal{I}_\Delta$  is a mapping from terminating terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F} \cup \{\perp, c\}, \mathcal{X})$  defined as follows:*

$$\mathcal{I}_\Delta(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ f(\mathcal{I}_\Delta(t_1), \dots, \mathcal{I}_\Delta(t_n)) & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \in \Delta \\ c(f(\mathcal{I}_\Delta(t_1), \dots, \mathcal{I}_\Delta(t_n)), t') & \text{if } t = f(t_1, \dots, t_n) \text{ and } f \notin \Delta \end{cases}$$

$$\text{where } t' = \text{order}(\{\mathcal{I}_\Delta(u) \mid t \rightarrow_{\mathcal{R}} u\})$$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{c}(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

In [GLU08], we extended this transformation in two novel ways:

- Extending it to deal with non- $\mu$ -terminating terms, which is not possible in the previous definition.

**Definition 91 ( $\mu$ -Interpretation [GLU08])** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\Delta \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathbf{c}\}, \mathcal{X})$  where  $\perp$  is a fresh constant symbol and  $\mathbf{c}$  is a fresh binary symbol (with  $\mu(\mathbf{c}) = \{1, 2\}$ ). The  $\mu$ -interpretation  $\mathcal{I}_{1,\Delta,\mu}$  is a mapping from arbitrary terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathbf{c}\}, \mathcal{X})$  defined as follows:

$$\mathcal{I}_{1,\Delta,\mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ \mathbf{f}(\mathcal{I}_{1,\Delta,\mu}(t_1), \dots, \mathcal{I}_{1,\Delta,\mu}(t_n)) & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \text{ and } \mathbf{f} \in \Delta \\ & \text{or } t \text{ is non-}\mu\text{-terminating} \\ \mathbf{c}(\mathbf{f}(\mathcal{I}_{1,\Delta,\mu}(t_1), \dots, \mathcal{I}_{1,\Delta,\mu}(t_n)), t') & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \text{ and } \mathbf{f} \notin \Delta \\ & \text{and } t \text{ is } \mu\text{-terminating} \end{cases}$$

$$\text{where } t' = \text{order}(\{\mathcal{I}_{1,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\})$$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{c}(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

- Extending it to be context-sensitive to obtain the refined notion of basic usable rule with strongly  $\mu$ -conservative cases.

**Definition 92 (Basic  $\mu$ -Interpretation [GLU08])** Let  $(\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mu}$  and  $\Delta \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathbf{c}\}, \mathcal{X})$  where  $\perp$  is a fresh constant symbol and  $\mathbf{c}$  is a fresh binary symbol. The basic  $\mu$ -interpretation  $\mathcal{I}_{2,\Delta,\mu}$  is a mapping from  $\mu$ -terminating terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathbf{c}\}, \mathcal{X})$  defined as follows:

$$\mathcal{I}_{2,\Delta,\mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ \mathbf{f}(\mathcal{I}_{2,\Delta,\mu,\mathbf{f},1}(t_1), \dots, \mathcal{I}_{2,\Delta,\mu,\mathbf{f},n}(t_n)) & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \\ & \text{and } \mathbf{f} \in \Delta \\ \mathbf{c}(\mathbf{f}(\mathcal{I}_{2,\Delta,\mu,\mathbf{f},1}(t_1), \dots, \mathcal{I}_{2,\Delta,\mu,\mathbf{f},n}(t_n)), t') & \text{if } t = \mathbf{f}(t_1, \dots, t_n) \\ & \text{and } \mathbf{f} \notin \Delta \end{cases}$$

$$\text{where } \mathcal{I}_{2,\Delta,\mu,\mathbf{f},i}(t) = \begin{cases} \mathcal{I}_{2,\Delta,\mu}(t) & \text{if } i \in \mu(\mathbf{f}) \\ t & \text{if } i \notin \mu(\mathbf{f}) \end{cases}$$

$$t' = \text{order}(\{\mathcal{I}_{2,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\})$$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{c}(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

In [AEF<sup>+</sup>08], the usable rules were introduced as part of the reduction pair processor for CSR.

**Remark 93** The  $\mu$ -reduction triple processor with usable rules is a good example to show that having a notion of chain that is apparently closer to the one for term rewriting (DP framework [GTSKF06, Thi08]), by definition, does not simplify the adaptation of processors to CSR. With regard to its use for proving termination of CSR, the set of usable rules has to be adapted if we use the notions of problem and chain in [AEF<sup>+</sup>08] or the notion of problem and chain in [GL10a], and it is different to the standard usable rules which are correct in the DP framework. ■

## 6.8 Subterm Criterion

In [HM04, HM07], Hirokawa and Middeldorp introduced a very interesting *subterm criterion* that permits certain cycles of the dependency graph to be ignored *without paying attention to the rules of the TRS*. Thiemann has adapted it to the DP-framework [Thi08, Section 4.6].

We start the adaptation of the subterm criterion to CSR with some definitions.

**Definition 94 (Root Symbols of a TRS [AGL10])** Let  $\mathcal{R}$  be a TRS. The set of root symbols associated to  $\mathcal{R}$  is:

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

**Definition 95 (Simple Projection [GL10b])** Let  $\mathcal{R}$  be a TRS. A simple projection for  $\mathcal{R}$  is a mapping  $\pi$  that assigns an argument position  $i \in \{1, \dots, k\}$  to every  $k$ -ary symbol  $f \in \text{Root}(\mathcal{R})$ . This mapping is extended to terms by

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

**Theorem 96 (Subterm Processor [GL10b])** 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)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , for all  $s \rightarrow t \in \mathcal{S}_{\#}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\#}\}$ . Let  $\mathcal{P}_{\pi, \triangleright_{\mu}} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright_{\mu} \pi(v)\}$  and  $\mathcal{S}_{\pi, \triangleright_{\mu}} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow t \in \mathcal{S}_{\#} \mid s \triangleright_{\mu} \pi(t)\}$ . Then,  $\text{Proc}_{\text{subterm}}$  given by

$$\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright_{\mu}}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright_{\mu}}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \triangleright_{\mu} \\ & \text{and whenever } \mathcal{P}_{\mathcal{X}} \neq \emptyset, \\ & \text{then } \mathcal{S}_{\pi} \subseteq \triangleright_{\mu} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Note that the conditions in Theorem 96 are not harmful in practice because the CS problems created from CS-TRSs fulfill those conditions.

---

**Example 97**

Consider the CS problems in Example 64, i.e.,

$$\begin{aligned}\tau_1 &= (\{(4.1)\}, \mathcal{R}, \emptyset, \mu^\sharp) \\ \tau_2 &= (\{(4.14)\}, \mathcal{R}, \emptyset, \mu^\sharp) \\ \tau_3 &= (\{(4.18)\}, \mathcal{R}, \emptyset, \mu^\sharp) \\ \tau_4 &= (\{(4.19)\}, \mathcal{R}, \emptyset, \mu^\sharp)\end{aligned}$$

We can conclude the finiteness of all of them by using the subterm criterion four times with the following (successive) projections:

$$\begin{aligned}\pi_1(+^\sharp) &= 1 \\ \pi_2(*^\sharp) &= 1 \\ \pi_3(\text{PRODOFFRACS}) &= 1 \\ \pi_4(\text{TAKE}) &= 1\end{aligned}$$

that is:

$$\begin{aligned}\pi_1(\mathbf{s}(n) +^\sharp m) = \mathbf{s}(n) &\triangleright_\mu n = \pi_1(n +^\sharp m) \\ \pi_2(\mathbf{s}(n) *^\sharp m) = \mathbf{s}(n) &\triangleright_\mu n = \pi_2(n *^\sharp m) \\ \pi_3(\text{PRODOFFRACS}(p :: ps)) = p :: ps &\triangleright_\mu ps = \pi_3(\text{PRODOFFRACS}(ps)) \\ \pi_4(\text{TAKE}(\mathbf{s}(n), x : xs)) = \mathbf{s}(n) &\triangleright_\mu n = \pi_4(\text{TAKE}(n, xs))\end{aligned}$$

---

**Example 98**

With regard to Example 65, the only CS problems left are:

$$\begin{aligned}\tau_1 &= (\{(4.24)\}, \mathcal{R}, \emptyset, \mu^\sharp) \\ \tau_2 &= (\{(4.25)\}, \mathcal{R}, \emptyset, \mu^\sharp)\end{aligned}$$

Both can be proved to be finite by using the subterm criterion twice with the following projections:

$$\begin{aligned}\pi_1(\geq^\sharp) &= 1 \\ \pi_2(-^\sharp) &= 1\end{aligned}$$

that is:

$$\begin{aligned}\pi_1(\mathbf{s}(x) \geq^\sharp \mathbf{s}(y)) = \mathbf{s}(x) &\triangleright_\mu x = \pi_1(x \geq^\sharp y) \\ \pi_2(\mathbf{s}(x) -^\sharp \mathbf{s}(y)) = \mathbf{s}(x) &\triangleright_\mu x = \pi_2(x -^\sharp y)\end{aligned}$$


---

**Example 99**

Consider  $\mathcal{R}$  and  $\mu$  as in Example 72, except for symbol `if` where we let  $\mu(\text{if}) = \{1\}$  instead of  $\mu(\text{if}) = \{1, 2\}$ . Now, consider  $\mu^\sharp$  in the usual way. The CSDPs  $\text{DP}(\mathcal{R}, \mu)$  are:

$$\mathbf{s}(x) >^\sharp \mathbf{s}(y) \rightarrow x >^\sharp y \quad (6.29)$$

$$x -^\sharp y \rightarrow y >^\sharp 0 \quad (6.30)$$

$$x -^\sharp y \rightarrow \text{IF}(y > 0, \mathbf{p}(x) - \mathbf{p}(y), x) \quad (6.31)$$

$$\mathbf{s}(x) \div^\sharp \mathbf{s}(y) \rightarrow (x - y) \div^\sharp \mathbf{s}(y) \quad (6.32)$$

$$\mathbf{s}(x) \div^\sharp \mathbf{s}(y) \rightarrow x -^\sharp y \quad (6.33)$$

$$\text{IF}(\text{true}, x, y) \rightarrow x \quad (6.34)$$

$$\text{IF}(\text{false}, x, y) \rightarrow y \quad (6.35)$$

The TRS  $\text{unh}(\mathcal{R}, \mu)$  is:

$$x - y \rightarrow x \quad (6.36)$$

$$x - y \rightarrow y \quad (6.37)$$

$$\mathbf{p}(x) - \mathbf{p}(y) \rightarrow \mathbf{p}(x) -^\sharp \mathbf{p}(y) \quad (6.38)$$

$$\mathbf{p}(x) \rightarrow \mathbf{P}(x) \quad (6.39)$$

If we consider the CS problem  $\tau_0 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$  and we apply  $\text{Proc}_{SCC}(\tau_0)$ , one of the resulting problems is  $\tau_1 = (\mathcal{P}_1, \mathcal{R}, \mathcal{S}_1, \mu^\sharp)$ , where  $\mathcal{P}_1 = \{(6.34), (6.35), (6.31)\}$  and  $\mathcal{S}_1 = \{(6.38), (6.36), (6.37)\}$ . By applying  $\text{Proc}_{UR}(\tau_1)$ , we can remove the pair (6.34) with the following polynomial interpretation:

$$\begin{array}{ll} [-^\sharp](x, y) = 2x + 2y + \frac{1}{2} & [\text{IF}](x, y, z) = \frac{1}{2}x + y + z \\ [-](x, y) = 2x + 2y + \frac{1}{2} & [>](x, y) = 2x + \frac{1}{2}y \\ [\text{if}](x, y, z) = \frac{1}{2}x + y + z & [\mathbf{p}](x) = \frac{1}{2}x \\ [\text{false}] = 0 & [\text{true}] = 2 \\ [0] = 0 & [\mathbf{s}](x) = 2x + 2 \end{array}$$

where all the rules are usable except the  $(\div)$ -rules. The resulting CS problem is  $\tau_2 = (\mathcal{P}_2, \mathcal{R}, \mathcal{S}_1, \mu^\sharp)$  where  $\mathcal{P}_2 = \{(6.35), (6.31)\}$ . By applying the subterm criterion, we can remove all rules in  $\mathcal{S}_1$  with the following projection for  $\mathcal{P}$ :

$$\begin{array}{ll} \pi(\text{IF}) & = 3 \\ \pi(-^\sharp) & = 1 \end{array}$$

that is:

$$\begin{aligned}
\pi(\text{IF}(\text{false}, x, y)) = y &\triangleright_{\mu} y = \pi(y) \\
\pi(x -^{\#} y) = x &\triangleright_{\mu} x = \pi(\text{IF}(y > 0, \mathfrak{p}(x) - \mathfrak{p}(y), x)) \\
\pi(\mathfrak{p}(x) - \mathfrak{p}(y)) = \mathfrak{p}(x) - \mathfrak{p}(y) &\triangleright_{\mu} \mathfrak{p}(x) = \pi(\mathfrak{p}(x) -^{\#} \mathfrak{p}(y)) \\
\pi(\mathfrak{p}(x) - \mathfrak{p}(y)) = \mathfrak{p}(x) - \mathfrak{p}(y) &\triangleright_{\mu} x = \pi(x) \\
\pi(\mathfrak{p}(x) - \mathfrak{p}(y)) = \mathfrak{p}(x) - \mathfrak{p}(y) &\triangleright_{\mu} y = \pi(y)
\end{aligned}$$

And obtaining the CS problem:

$$\tau_3 = (\{(6.35), (6.31)\}, \mathcal{R}, \emptyset, \mu^{\#})$$

which is proved to be finite by using<sup>2</sup>  $\text{Proc}_{SCC}(\tau_3)$ .

Now we present a variant of the subterm criterion that is based on considering the frozen positions only. This processor comes from the adaptation of the technique developed in [AGL06, Theorem 6]. In contrast to the subterm processor, arbitrary (stable) quasi-orderings can be used. As in the subterm processor, we do not need to consider rules in  $\mathcal{R}$ , but only the rules in  $\mathcal{P} \cup \mathcal{S}$ .

**Theorem 100 (Non- $\mu$ -Replacing Projection Processor [GL10b])** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem where  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$ . Assume that (1)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , for all  $s \rightarrow t \in \mathcal{S}_{\#}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\#}\}$ . Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded such that*

1. for all  $f \in \text{Root}(\mathcal{P})$ ,  $\pi(f) \notin \mu(f)$ ,
2.  $\pi(\mathcal{P}) \subseteq \succsim$ , and,
3. whenever  $\mathcal{S} \neq \emptyset$  and  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , we have that  $\mathcal{S}_{\pi} \subseteq \succsim$

Let  $\mathcal{P}_{\pi, >} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$  and  $\mathcal{S}_{\pi, >} = \{s \rightarrow t \in \mathcal{S}_{\triangleright_{\mu}} \mid s > t\} \cup \{s \rightarrow t \in \mathcal{S}_{\#} \mid s > \pi(t)\}$ . Then, the processor  $\text{Proc}_{NRP}$  given by

$$\text{Proc}_{NRP}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, >}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, >}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

<sup>2</sup>The complete automatic proof of this example can be found in <http://zenon.dsic.upv.es/muterm/> in the CSR benchmarks (example called TRS/aprove08-csr/csrdiv.tris).

**Example 101**

Consider the following TRS  $\mathcal{R}$  [Zan97, Example 1]:

$$\begin{aligned} g(x) &\rightarrow h(x) \\ c &\rightarrow d \\ h(d) &\rightarrow g(c) \end{aligned}$$

together with  $\mu(g) = \mu(h) = \emptyset$ . Note that  $\mathcal{R}$  is  $\mu$ -conservative. Now,  $\text{DP}(\mathcal{R}, \mu)$  consists of the following (noncollapsing) CSDPs:

$$\begin{aligned} G(x) &\rightarrow H(x) \\ H(d) &\rightarrow G(c) \end{aligned}$$

and  $\text{unh}(\mathcal{R}, \mu)$  is:

$$c \rightarrow C$$

Then, for the CS problem  $\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ , we can apply  $\text{Proc}_{NRP}$  to remove the second pair in  $\tau$  by using the following projection<sup>3</sup>:

$$\begin{aligned} \pi(G) &= 1 \\ \pi(H) &= 1 \end{aligned}$$

and the following polynomial interpretation:

$$\begin{aligned} [d] &= 1 & [c] &= 0 \\ \pi(G(x)) = x &\geq x = \pi(H(x)) \\ \pi(H(d)) = d &> c = \pi(G(c)) \end{aligned}$$

We can conclude finiteness of the resulting problem using the SCC processor.

---

### 6.8.1 Historical Development of Subterm Criterion for CSR

The subterm criterion is one of the fastest and most successful processors in the CSDP framework. The subterm criterion for CSR was first defined in [AGL06], but it is limited to use with cycles  $\mathfrak{C}$  including noncollapsing pairs only, i.e.,  $\mathfrak{C} \subseteq \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ . A similar technique based on using stable quasi-orderings to compare pairs (but without paying attention to the rules in  $\mathcal{R}$  as in the subterm criterion) was introduced and proved correct under certain conditions. In [AGL10], these techniques were defined as processors. A new

<sup>3</sup>Since  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$ , we do not need to satisfy (3) in Theorem 100.

---

version of the subterm criterion was then defined for collapsing pairs. Finally, in [GL10a], thanks to the flexibility of using a new TRS  $\mathcal{S}$ , we could adapt the classical subterm criterion to be used with both collapsing and noncollapsing pairs, thereby allowing us to remove individual rules from  $\mathcal{S}$  (see Theorems 96 and 100).



---

# 7

## Implementing the CSDP Framework

The CSDP framework described in this thesis has been implemented as part of the tool MU-TERM, a tool for proving termination properties. The system has three main parts: the *input/output interface*, the *framework*, and the *constraint solver*.

Nowadays, we have a command line input/output interface and a web interface. MU-TERM accepts CS-TRSs written in the TPDB format<sup>1</sup> and returns a detailed proof in plain text. The web page of the tool is the following:

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

Proofs of termination in MU-TERM rely heavily on the generation of polynomial orderings using polynomial interpretations with rational coefficients [Luc05] and matrix interpretations with rational entries [ALNM09]. The constraint solver subsystem is a key component of the tool. It is based on the SMT-based constraint-solver for rational numbers reported in [BLNM<sup>+</sup>09].

The termination framework implements the processors described in this thesis. In this chapter, we pay attention to the implementation of this termination framework part.

### 7.1 The Language

Our tool has been implemented in the functional language Haskell [Jon03]. We find it appropriate to develop a robust and large project such as MU-TERM where the number of participants is reduced. Haskell is an advanced, purely functional programming language and an open source product that incorporates more than twenty years of cutting-edge research in the area. Haskell is

---

<sup>1</sup>see <http://www.lri.fr/~marche/tpdb/format.html>

appropriate for the rapid development of robust, concise, and correct software. With strong support for integration with other languages, built-in concurrency and parallelism, debuggers, profilers, rich libraries and an active community, Haskell makes it easy to produce flexible, maintainable, high-quality software<sup>2</sup>.

For further information about Haskell you can read [Bir98, OGS08].

## 7.2 Term Rewriting Systems

In this thesis, we only consider CS-TRSs, but MU-TERM is designed to work with different kinds of TRSs: term rewriting systems, context-sensitive TRSs, order-sorted term rewriting systems, associative and/or commutative term rewriting systems, etc. Depending on the system considered, some data related to the signature or the variables (sort information, arity, replacement map...) must be stored. An essential ingredient for obtaining a robust definition of TRS is to make the terms parametric on the two kinds of symbols that can be used to build them: function symbols and variable symbols. Then, terms are defined as a recursive abstract data type with two constructors.

```
data Term idF idV = F idF [Term idF idV]
                  | V idV
```

This allows us to obtain rules and TRSs that are parametric on the type of the identifiers.

We use typeclasses [OGS08, Chapter 6] to develop specific functions that depend on the type of the identifier. Typeclasses allow us to abstract about the concrete definitions of an identifier type and focus on its features instead. For instance, we have typeclasses (as `HasArity` or `HasMu`) that give us access to the arity or replacement map of an identifier. Only those types that have an associated replacement map can instantiate the `HasMu` class.

```
class HasArity a where
  getFArity  :: a -> Int
  setFArity  :: Int -> a -> a

class HasMu a where
  getFMu  :: a -> [Int]
  setFMu  :: [Int] -> a -> a
```

---

<sup>2</sup><http://www.haskell.org/>

With this information, we can define the different kind of systems which are handled by MU-TERM. Furthermore, we can easily combine features from different kinds of systems to obtain new ones without modifying our functions. For example, we define function symbols as records with the following information:

```
data IdFTRS = IdFTRS
  { fid      :: Int
  , fname    :: String
  , arity    :: Int
  }
```

That is, a function symbol has an identifier, a name and an arity. We can define an instance of `HasArity` for this symbol:

```
instance HasArity IdFTRS where
  getFArity = arity
  setFArity ar (IdFTRS { fid  = fid'
                       , fname = fname'
                       })
    = IdFTRS { fid  = fid'
             , fname = fname'
             , arity = ar
             }
```

Furthermore, we can define function symbols with replacement restrictions by using an `IdFTRS` function symbol and defining instances of `HasArity` and `HasMu` for it:

```
data IdFCS id = IdFCS
  { csfprev  :: id
  , mu       :: [Int]
  } deriving Show

instance (HasArity prev) => HasArity (IdFCS prev) where
  getFArity  = getFArity . csfprev
  setFArity ar = fmap (setFArity ar)

instance HasMu (IdFCS prev) where
  getFMu = mu
  setFMu newmu (IdFCS { csfprev = csfprev
                       , mu      = csmu })
```

```

= IdFCS { csfprev = csprev
        , mu = newmu
        }

```

Then, the type of our context-sensitive function symbol is `IdFCS IdFTRS`.

## 7.3 CSDP Framework

The CSDP framework consists mainly of four components: the problems, the proofs, the CS processors, and the strategy that is used to apply and combine the different CS processors to explore the search tree that is mentioned in Theorem 51.

### 7.3.1 Problems

Thus, the idea is to translate the developments in Chapter 5 to our implementation. CS problems are just triples that contain the TRSs  $\mathcal{P}$ ,  $\mathcal{R}$ , and  $\mathcal{S}$  (the information about the replacement map is encoded as part of the description of the identifiers). As mentioned in Section 7.2, MU-TERM implements different frameworks (DP framework, CSDP framework, A $\forall$ C framework...) and we have different kinds of problems. A CS problem is defined in the following way:

```

data CS = CS

instance IsProblem CS where
  data Problem CS trs = CProb trs trs trs
  getProblemType (CProb _ _ _) = CS
  getR (CProb r _ _) = r
  getP (CProb _ p _) = p
  getS (CProb _ _ s) = s

```

To create CS problems or to modify their information, we use a different type-class:

```

instance MkCSProblem CS trs where
  mkCSProblem CS r p s = CProb r p s
  mapR f (CProb r p s) = CProb (f r) p s
  mapP f (CProb r p s) = CProb r (f p) s
  mapS f (CProb r p s) = CProb r p (f s)

```

Note that when we treat CS problems, the variable `trs` is instantiated to `TRS idF idV (Term idF idV)` where the identifier `idF` is `IdFCS IdFTRS`, which is an instance of `HasAriety` and `HasMu`.

### 7.3.2 Proof

In the CSDP framework, when you apply a processor to a problem, the result is a list of problems. We construct a proof tree by the iterative application of processors to a problem until either a solution is reached or the search space is exhausted. The intermediate nodes of a proof tree keep track of the applied processors. We consider four kinds of intermediate nodes in a proof tree.

- A list of problems with the constructor `And`.
- An empty list of problems with the constructor `Success`.
- A refutation of the finiteness of the problem with the constructor `Refuted`.
- A `DontKnow` response.

All these alternatives are returned along with the information of the applied processor (`procInfo`).

```
data Proof a =
  And      {procInfo, inProblem :: !(SomeInfo)
            , subProblems :: [Proof a]}
| Success {procInfo, inProblem :: !(SomeInfo)}
| Refuted {procInfo, inProblem :: !(SomeInfo)}
| DontKnow {procInfo, inProblem :: !(SomeInfo)}
| Return a
```

The constructor `Return` corresponds to the leaves of the tree and stores the returned CS problems. The data type `Proof` is a Monad [Awo06]. We use `SomeInfo` as a super-type that encapsulates all kinds of problems and information, obtaining a homogeneous type.

```
data SomeInfo where
  SomeInfo :: p -> SomeInfo

someInfo :: p -> SomeInfo
someInfo  = SomeInfo
```

### 7.3.3 Processors

Our processors get an input problem and return a `Proof` where its leaves are problems. The following typeclass models this notion (specifying the particular processor to be applied).

```
class Processor name problem where
  apply :: name -> problem -> Proof problem
```

For example, in order to implement the basic processors in Section 6.3, first we define the name `BasicProcessor`. We also have to define the resulting information of the application of the processor. In this case:

- `BasicProcInfo` tells us that all the pairs are pairs in  $\mathcal{P}_X^1$  and they satisfy the conditions of the processor (see Theorem 73).
- `BasicRefutedProcInfo` tells us that there is pair in cycle (see Theorem 71).
- otherwise, `BasicDontKnow` is returned.

```
data BasicProcessor = BasicProcessor
```

```
data BasicProcInfo problem idF idV
  = BasicProcInfo { inProblem :: problem }
  | BasicRefutedProcInfo
      { inProblem :: problem
      , inCycles  :: Set (Rule (Term idF idV))
      }
  | BasicDontKnow { inProblem :: problem }
```

The implementation of the processor is the following:

```
-- Processor
instance ( trs ~ TRS idF idV (Term idF idV)
        , HasAarity idF, HasMu idF
        , Ord idF, Ord idV
        , MkCSProblem CS trs
        ) => Processor BasicProcessor
      (Problem CS trs) where
  apply BasicProcessor inP
    = if (and . map isPx1) dps then
      success (BasicProcInfo inP) inP
```

```

else
  if (not . null) procInf then
    refuted (BasicRefutedProcInfo inP procInf) inP
  else
    dontKnow (BasicDontKnow inP) idF idV) inP
where trs = getR inP
      dps = getTRSRules . getP
      procInf
          = fromList [ l :-> r | l :-> r <- elems dps
                      , matches l (fresh r)]

```

The header tell us that the processor is applied to `Problem CS trs` problems and the identifiers of the TRSs are instances of `HasArity` and `HasMu`. The first condition (and `. map isDPx1`) `dps` checks whether all CSDPs belong to  $\mathcal{P}_{\mathcal{X}}^1$ ; the second condition checks whether there is a pair in cycle (`not . null`) `procInf`; finally, if the previous conditions are not satisfied, the processor returns `DontKnow`. Functions `success`, `refuted` and `dontKnow` create nodes `Success`, `Refuted` and `DontKnow`.

### 7.3.4 Strategy

One of the most important objectives of our implementation is to provide a strategy language for our CSDP framework. The language must be flexible to permit different combinators and, at the same time, robust to allow only correct strategies. A termination tool which incorporates a strategy language is  $\mathbb{T}_1\mathbb{T}_2$  [KSZM09]. A strategy takes a problem as input and returns a proof tree. The definition of a good strategy is not trivial. For this reason, we need flexible combinators to make the process of finding a good strategy easy.

In our implementation, our strategy language is Haskell itself. In this way, we have a typed strategy language and, on the other hand, the possibility of defining as many combinators as desired. For instance, our combinator to put two processors in a sequence, is the following.

```

(&.) :: (problem -> Proof problem)
      -> (problem -> Proof problem)
      -> problem -> Proof problem
(&.) = (>=>)

```

We can also develop more complex combinators, as the one which recursively applies a given strategy.

```

fixSolver :: (problem -> Proof problem)

```

```

-> problem -> Proof problem
fixSolver f x = let x' = f x in (x' >>= fixSolver f)

```

Of course, our processors must be terminating for using this combinator. In the same way, we have other combinators:

- a combinator (`.|. .`) that adds a decision point,
- a combinator that tries a given processor and if it fails, then continues with the strategy,
- a combinator that applies a processor recursively as long as it does not fail,

and so on. Another advantage of having the full language as the strategy language is that you can have parallel combinators or combinators with timeouts.

### Example 102

The following code is the example of a short strategy using the `.|. .` and `.&. .` combinators.

```

csdpStrat = sccProcessor
  .&. (subtermProc
    .|. (rtProc [I 0,I 1,I 2])
    .|. (rtProc [I 0,Q 1 2,I 1])
    .|. narrProc
  )
  .&. sccProcessor
  .&. fixSolver((subtermProc
    .|. (rtProc [I 0,I 1,I 2])
    .|. (rtProc [I 0,Q 1 2,I 1])
  )
  .&. sccProcessor
)

```

In the example, we first apply the SCC processor; then, we apply four different processors (subterm processor, RT processor using the values  $[0, 1, 2]$ , RT processor using the values  $[0, \frac{1}{2}, 1]$  and narrowing processor [AGL10]). This could be done in parallel. If all these processors fail, then the proof tree returns `DontKnow`. If one of them returns a new CS problem, then we apply the SCC processor and recursively the same processors (except the narrowing processor to avoid an infinite proof tree) until a success proof or a fail of the three processors in the same iteration is reached.

---

# 8

## Experiments

MU-TERM is a tool which can be used to verify a number of termination properties of (variants of) Term Rewriting Systems (TRSs): termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and termination of rewriting modulo specific axioms. Such termination properties are essential to prove termination of programs in sophisticated rewriting-based programming languages. Specific methods have been developed and implemented in MU-TERM in order to efficiently deal with most of them.

The implementation in MU-TERM [AGLNM10] of our CSDP framework has improved the state of the art of termination of CSR and the International Termination Competition<sup>1</sup> is a good gauge for checking this evolution. In 2006, AProVE won the CSR category in the International Termination Competition, proving 60 of 90 examples using transformations. Nowadays, MU-TERM can prove 95 of the current 109 examples stored in the TPDB<sup>2</sup> for the CSR category. We summarize the evolution of our experimental results in the following sections.

### 8.1 2007 Termination Competition

MU-TERM participated in the CSR subcategory of the 2007 International Termination Competition:

<http://www.lri.fr/~marche/termination-competition/2007>

The benchmarks were executed in a completely automatic way with a timeout of 60 seconds over the complete collection of 90 CSR systems of the Termination

---

<sup>1</sup><http://www.lri.fr/~marche/termination-competition/>

<sup>2</sup><http://www.lri.fr/~marche/tpdb>

Problem Data Base (TPDB, version 4.0). The results are summarized in Table 8.1

Tool	Proved	YES Average Time
MU-TERM 4.4	68/90	2.87s
AProVE 07	64/90	6.90s

Table 8.1: 2007 International Termination Competition results

As mentioned in Subsection 1.2.2, several methods have been developed to prove termination of CSR for a given CS-TRS  $(\mathcal{R}, \mu)$ . Two main approaches have been investigated so far:  $\mu$ -reduction orderings and transformations. Besides MU-TERM, AProVE was the only tool that is able to prove termination of CSR by using (non-trivial) transformations. In 2007, AProVE was the most powerful tool for proving termination of TRSs and most existing results and techniques regarding DPs and related techniques were implemented in AProVE. AProVE implemented a termination expert that successively tried different transformations for proving termination of CSR and used a variety of different and complementary techniques for proving termination of rewriting, see [GTSKF04, GTSK04]. MU-TERM was based on the newborn CSDP approach in [AGL06] with the improvements presented in [AGL07] and was capable of solving more examples and was faster than AProVE, showing that CSDPs were the most powerful and fastest technique for proving termination of CSR.

## 8.2 2009 Termination Competition

In December 2009, MU-TERM participated in the International Termination Competition again. The benchmarks were executed in a completely automatic way with a timeout of 60 seconds over a subset of 37 systems<sup>3</sup> of the complete collection of the 109 CS-TRSs of the TPDB 7.0.

The CSDP framework described in this thesis was implemented as part of the termination tool MU-TERM. The results of the competition are summarized in Table 8.2. The tools AProVE [GSKT06] and VMTL [SG09], implement the CSDPs using the transformational approach in [AEF<sup>+</sup>08]. To our knowledge, the techniques implemented by Jambox [End09] to prove termination of CSR have not yet been documented. As Table 8.2 shows, we are able to prove the same number of systems as AProVE, but MU-TERM is almost two 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

<sup>3</sup>See <http://termcomp.uibk.ac.at/>

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

Table 8.2: 2009 International Termination Competition results

from this collection of problems. And, as remarked in [GL10a], there are interesting examples that can only be handled by MU-TERM.

### 8.3 CS Processor Evaluation

Table 8.3 shows the use of the different processors over the 95 solved problems of the TPDB. The interpretation of the frequency of use for the different

Processors	Applied in
SCC Processor (Section 6.2)	95/95 problems
Processors based on subterm criteria (Section 6.8)	56/95 problems
Processors based on reduction triples (Section 6.6)	50/95 problems
Basic Processors (Section 6.3)	28/95 problems

Table 8.3: Summary of processors used in MU-TERM

processors should take into account the following *strategy* for applying them in MU-TERM when CS problems are treated: first, we try the basic (infinite and finite) processors. If some of them succeed, we are done; otherwise, we continue as follows [AGLNM10]:

**Definition 103 (Strategy of MU-TERM 5.0)** *The strategy used in MU-TERM to find a proof is:*

1. We check the system for extra variables at active positions in the right-hand sides of the rules.
2. We obtain the CSDPs and the unhiding TRS, building the initial CSDP problem. And now, recursively:
  - (a) Decision point between the Basic Processors and the SCC processor.
  - (b) Subterm criterion processor.
  - (c) Reduction triple (RT) processor with linear polynomials (LPoly) and coefficients in  $N_2 = \{0, 1, 2\}$ .

- (d) RT processor with LPoly and coefficients in  $Q_2 = \{0, 1, 2, \frac{1}{2}\}$  and  $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{1}{4}\}$  (in this order).
- (e) RT processor with simple mixed polynomials (SMPoly) and coefficients in  $N_2$ .
- (f) RT processor with SMPoly and rational coefficients in  $Q_2$ .
- (g) RT processor with 2-square matrices with entries in  $N_2$  and  $Q_2$ .
- (h) Transformation processors (only twice to avoid nontermination of the strategy): instantiation, forward instantiation, and narrowing.

3. If the techniques above fail, then we use CS-RPO.

Interestingly, *all* processors are used at least once during the proofs.

The arbitrary application of CS processor can generate an infinite search space. To avoid this infinite search space and to generate a finite proof tree in all the cases, we proceed in following way:

- Processors that remove pairs as RT processors return DontKnow when it returns the same set of pairs. In this way we decrease the size of the proof tree.
- Transformation processors are applied only a finite number of times.
- SCC processor continues been applied meanwhile a transformation processor is applied or a processor that remove pairs returns a smaller (in number) set of pairs.

Furthermore, we choose the criterion to decide the order of application of the CS processors basing on the speed of the CS processors. Then, we apply first fast CS processors that remove rules as subterm criterion and RTs with bounds (depending on the bound the speed of the processor changes) and later the transformation processors.

## 8.4 Contributions of the CSDP Framework to MU-TERM

In order to have some experimental evidence, we have also executed the complete collection of systems of the CSR category<sup>4</sup>, where we compare the first version of MU-TERM that uses CSDPs in 2006 [AGL06] (MU-TERM 4.3),

<sup>4</sup>A complete report of our experiments can be found in <http://zenon.dsic.upv.es/muterm/benchmarks/>

the version of MU-TERM used in the 2007 international termination competition [AGIL07] (MU-TERM 4.4), and the version of MU-TERM used in the 2009 international competition [AGLNM10] (MU-TERM 5.0). The results are shown in Table 8.4. In versions 4.3 and 4.4, the CSDP framework was not available.

Tool	Proved	YES Average Time
MU-TERM 4.3	65/109	3.37s
MU-TERM 4.4	80/109	1.07s
MU-TERM 5.0	95/109	1.13s

Table 8.4: Comparison among MU-TERM versions

Now, we can prove 30 more examples than the version 4.3 and 15 more examples than the version 4.4. Comparing the execution times between versions 4.4 and 5.0 over the 80 examples where both tools succeeded (84, 48 seconds vs. 15,073 seconds), we are more than 5,5 times faster in our last version. Actually, all the examples that were solved by using other techniques such as CSRPO, polynomial orderings, or transformational approaches were also solved using CSDPs.

## 8.5 Advantages of the CSDP Framework over Previous Approaches

The following Maude program combines the use of an evaluation strategy and types given as sorts in the specification [DLM<sup>+</sup>08].

```
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
```

We can use the transformation developed in [DLM<sup>+</sup>08] to transform this system into a CS-TRS (without sorts). Such a CS-TRS can be found in the Termination Problems Data Base<sup>5</sup> (TPDB): TRS/CSR\_Maude/LengthOfFiniteLists\_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 thesis<sup>6</sup>.

---

<sup>5</sup><http://www.lri.fr/~marche/tpdb/>

<sup>6</sup>On Sep 09, 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>

---

## Conclusions

We have shown that the investigation of the structure of infinite context-sensitive rewrite sequences and the characterization of minimal non- $\mu$ -terminating terms is crucial in order to understand the non- $\mu$ -terminating behavior of context-sensitive term rewriting systems. This knowledge is used to provide an appropriate definition of *context-sensitive dependency pair* and the related notion of *chain*. In sharp contrast to the dependency pair approach for term rewriting systems, where all the dependency pairs have tuple symbols  $F$  both in the left- and right-hand sides, we have *collapsing* pairs that have a single *variable* in the right-hand side. These variables reflect the effect of the *migrating* variables in the termination behavior of context-sensitive rewriting. At the level of *minimal chains*, however, this contrast is recovered by a nice symmetry that arises from the central notion of *hidden term* and *hiding context*: the right-hand sides that are discarded when the context-sensitive dependency pairs are computed now become crucial to know whether collapsing pairs involve infinite computations. With the notion of *hidden term* and *hiding context*, we can construct an *unhiding term rewriting system* that simulates the minimal steps in  $\mu$ -rewrite sequences, and we obtain a simple and expressive notion of chain. As in Arts and Giesl's approach, the absence of infinite chains of context-sensitive dependency pairs characterizes the  $\mu$ -termination of  $\mathcal{R}$ .

We have provided a suitable adaptation of Giesl et al.'s *dependency pair framework* to context-sensitive rewriting by defining appropriate notions of *context-sensitive problem* and *context-sensitive processor*. In this setting, we have described a number of sound and (most of them) complete context-sensitive processors that can be used in any practical implementation of the *context-sensitive dependency pair framework*.

We have implemented these ideas as part of the MU-TERM termination tool. The implementation and practical use of the developed techniques yield a novel and powerful framework that dramatically improves the current state-of-the-art of methods for proving termination of context-sensitive rewriting. In fact, context-sensitive dependency pairs were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007, 2009, and 2010 competitions of termination tools, and previous techniques such as transformations and  $\mu$ -reduction orderings based on polynomial interpretations and

path orderings are currently obsolete in the area.

With regard to future work, we think interesting to address the following issues and problems:

- Exploit the presence of the component  $\mathcal{S}$  in the definition of context-sensitive problem by defining new processors that specifically take into account this component. In this setting, another interesting topic of research is the development of techniques for generating the  $\mu$ -reduction triples that are used in the  $\mu$ -reduction triple processor.
- Improve the estimation of the context-sensitive graph by using the ideas developed in [Mid02], where tree automata techniques are used to provide more precise estimations to the reachability problems that are involved in the definition of the (dependency) graph.
- Integrate the use of argument filterings into the processor of  $\mu$ -reduction triples with usable rules.
- Obtain a modular analysis of context-sensitive rewriting using context-sensitive dependency pairs. Modularity issues for context-sensitive rewriting were investigated for the first time in [GL02a, GL02b]. We plan to consider Urbain's *rewriting modules* [Urb04] and the corresponding analysis of modularity of termination to extend previous results about modularity of termination of context-sensitive rewriting.
- Proofs of non-termination of context-sensitive rewriting. Loops in context-sensitive computations have been recently investigated by Thiemann and Sternagel [TS09]. Their results have been used to furnish the last version of MU-TERM with a more powerful processor for detecting non-termination of context-sensitive rewriting (see [AGLNM10]). We plan to use our experience in the analysis of infinite  $\mu$ -rewrite sequences to obtain more advanced non- $\mu$ -termination processors.
- Compare *monotonic semantic path ordering* [BFR00] and our context-sensitive dependency pair framework. It is well-known that dependency pairs can be seen as a special case of the monotonic semantic path ordering approach to termination of rewriting. In her PhD thesis, Borralleras has proved that monotonic semantic path ordering can be adapted to prove termination of context-sensitive rewriting [Bor03]. From a theoretical point of view, it is interesting to explore how context-sensitive dependency pairs are connected to Borralleras' context-sensitive monotonic semantic path ordering.

- Certification of termination proofs for context-sensitive term rewriting systems. This is a hot topic of research. Due to its complexity and size, the development of automatic termination tools is an error prone activity which often leads to buggy implementations which can produce wrong proofs. For this reason, the development of tools that are able to use proof assistants like Coq or Isabelle to automatically check that the outcome of a termination tool is a valid proof of termination appears as a highly desirable complement to termination tools. With regard to context-sensitive rewriting, no appropriate libraries or Coq/Isabelle definitions of specific techniques for proving termination of context-sensitive rewriting have been developed yet.



---

# Conclusiones

Hemos mostrado que el estudio de la estructura de las secuencias de reescritura sensibles al contexto infinitas y la caracterización de los términos no- $\mu$ -terminantes son cruciales para entender el comportamiento no- $\mu$ -terminante de los sistemas de reescritura sensibles al contexto. Esta información se ha usado para obtener una definición apropiada de par y de cadena de dependencia sensible al contexto. En contraposición al de los pares de dependencia en reescritura, donde todos los pares están encabezados tanto en su parte izquierda como en su parte derecha por un símbolo tupla  $F$ , tenemos pares de dependencia *colapsantes*, cuya parte derecha es una *variable*. Estas variables reflejan el efecto de las variables *migrantes* in el comportamiento terminante de la reescritura sensible al contexto. Sin embargo, a nivel de *cadena minimal*, este contraste es recuperado por la bella simetría obtenida por las nociones de *término oculto* y *contexto ocultador*: las partes derechas que son desechadas cuando se obtienen los pares de dependencia son cruciales para saber si los pares colapsantes generan computaciones infinitas o no. Con las nociones de término oculto y contexto ocultador podemos construir un *sistema de reescritura revelador*, que simula los pasos minimales en las secuencias de  $\mu$ -reescritura y nos permite conseguir una noción de cadena simple pero efectiva. Como en la aproximación de Arts y Giesl, la presencia o ausencia de cadenas infinitas de pares de dependencia caracterizan la  $\mu$ -terminación de  $\mathcal{R}$ .

Hemos adaptado el *marco de pares de dependencia* de Giesl et al. a la reescritura sensible al contexto proporcionando una correcta definición de *problema sensible al contexto* y *procesador sensible al contexto*. Con esta configuración, hemos descrito una serie de procesadores sensibles al contexto correctos y (la mayoría de ellos) completos que pueden ser usados de forma práctica en cualquier implementación de un marco de pares de dependencia sensibles al contexto.

Hemos implementado estas ideas como parte de la herramienta de terminación MU-TERM. La implementación y los resultados prácticos de las técnicas desarrolladas desembocan en un potente y novedoso marco que mejora de forma considerable el actual estado del arte de los métodos para probar la terminación de la reescritura sensible al contexto. De hecho, los pares de dependencia sensibles al contexto son un ingrediente fundamental para que MU-TERM haya sido capaz de ganar la competición de terminación en la cate-

goría de reescritura sensible al contexto en los años 2007, 2009 y 2010. Además, técnicas previas como transformaciones, órdenes de  $\mu$ -reducción basados en interpretaciones polinómicas y órdenes de caminos están obsoletos en el área. Así es que podemos decir que los pares de dependencia sensibles al contexto son la base de las herramientas de terminación que prueban la terminación de la reescritura sensible al contexto.

Como trabajo futuro estamos interesados en abordar las siguientes tareas y problemas:

- Aprovechar la presencia del componente  $\mathcal{S}$  en la definición de problema sensible al contexto para definir nuevos procesadores que tengan en cuenta este componente de forma específica. Sobre esta configuración, otro tema interesante de investigación es el desarrollo de técnicas para generar triples de  $\mu$ -reducción que son utilizados en el procesador de triples de  $\mu$ -reducción.
- Mejorar la estimación del grafo de dependencia usando las técnicas desarrolladas en [Mid02], donde las técnicas basadas en autómatas de árboles se usan para obtener una estimación más precisa del problema derivado de la alcanzabilidad en la definición de grafo de dependencia.
- Integrar el uso del filtrado de argumentos en el procesador de triples de  $\mu$ -reducción con reglas usables.
- Obtener un análisis modular de la reescritura sensible al contexto usando pares de dependencia sensibles al contexto. La modularidad de la reescritura sensible al contexto fue investigada por primera vez en [GL02a, GL02b]. Nuestro plan es considerar los *módulos de reescritura* de Urbain [Urb04] y su análisis de la modularidad de la terminación para extender sus resultados a la reescritura sensible al contexto.
- Probar no-terminación de la reescritura sensible al contexto. Recientemente, Thiemann y Sternagel [TS09] han investigado los bucles en las computaciones sensibles al contexto. Sus resultados han sido utilizados para mejorar la última versión de MU-TERM con un procesador para la detección de no-terminación de la reescritura sensible al contexto más potente (ver [AGLNM10]). Planificamos usar nuestra experiencia en el análisis de las secuencias de  $\mu$ -reescritura para conseguir un procesador de no- $\mu$ -terminación más avanzado.
- Comparar los *órdenes de caminos semánticos monótonos* [BFR00] (sensibles al contexto) y nuestro marco de pares de dependencia sensibles al

contexto. Es bien conocido que los pares de dependencia se pueden ver como un caso especial de orden de caminos semánticos monótonos para la terminación de la reescritura. En su tesis doctoral, Borralleras demostró que el orden de caminos semánticos monótonos se puede adaptar para probar la terminación de la reescritura sensible al contexto [Bor03]. Desde un punto de vista teórico, es interesante explorar cómo los pares de dependencia sensibles al contexto están conectados con el orden de caminos semánticos monótonos sensible al contexto de Borralleras.

- Certificar las pruebas de terminación para los sistemas de reescritura sensibles al contexto, que actualmente es un tema de investigación candente. Debido a su complejidad y tamaño, el desarrollo de herramientas de terminación puede contener errores implementación que a menudo conllevan errores en las pruebas de terminación. Por esta razón, el desarrollo de herramientas que puedan usar asistentes de pruebas como Coq o Isabelle, que pueden validar que la salida de la herramienta de terminación es una prueba correcta, aparece como un complemento altamente deseable en las herramientas de terminación. Con respecto a la reescritura sensible al contexto, todavía no se han desarrollado bibliotecas apropiadas ni definiciones de Coq/Isabelle de técnicas específicas para probar la terminación de la reescritura sensible al contexto.



---

## Conclusions

Hem mostrat que l'estudi de l'estructura de les seqüències de reescriptura sensibles al context infinites i la caracterització dels termes no- $\mu$ -terminants són crucials per a entendre el comportament no- $\mu$ -terminant dels sistemes de reescriptura sensibles al context. Aquesta informació s'ha usat per a obtenir una definició apropiada de parell i de cadena de dependència sensible al context. En contraposició al dels parells de dependència en reescriptura, on tots els parells estan encapçalats tant a la seua part esquerra com a la seua part dreta per un símbol tupla  $F$ , tenim parells de dependència *col.lapsants*, la part dreta de la qual és una *variable*. Aquestes variables reflecteixen l'efecte de les variables *migrants* en el comportament terminant de la reescriptura sensible al context. No obstant això, a nivell de *cadena minimal*, aquest contrast és recuperat per la bella simetria obtinguda per les nocions de *terme ocult* i *context ocultador*: les parts dretes que són rebutjades quan s'obtenen els parells de dependència són crucials per a saber si els parells colapsants generen computacions infinites o no. Amb les nocions de terme ocult i context ocultador podem construir un *sistema de reescriptura revelador*, que simula els passos minimal en les seqüències de  $\mu$ -reescriptura i ens permet aconseguir una noció de cadena simple però efectiva. Com en l'aproximació de Arts i Giesl, la presència o absència de cadenes infinites de parells de dependència caracteritzen la  $\mu$ -terminació de  $\mathcal{R}$ .

Hem adaptat el *marc de parells de dependència* de Giesl et al. a la reescriptura sensible al context proporcionant una correcta definició de *problema sensible al context* i *processador sensible al context*. Amb aquesta configuració, hem descrit una sèrie de processadors sensibles al context correctes i (la majoria d'ells) complets que poden ser usats de forma pràctica en qualsevol implementació d'un marc de parells de dependència sensibles al context.

Hem implementat aquestes idees com part de l'eina de terminació MU-TERM. La implementació i els resultats pràctics de les tècniques desenvolupades desemboquen en un potent i nou marc que millora de forma considerable l'actual estat de l'art dels mètodes per a provar la terminació de la reescriptura sensible al context. De fet, els parells de dependència sensibles al context són un ingredient fonamental perquè MU-TERM haja estat capaç de guanyar la competició de terminació en la categoria de reescriptura sensible al context en els anys 2007, 2009 i 2010. A més, tècniques prèvies com transformacions, ordres

de  $\mu$ -reducció basats en interpretacions polinòmiques i ordres de camins estan obsolets en l'àrea. Així és que podem dir que els parells de dependència sensibles al context són la base de les eines de terminació que proven la terminació de la reescriptura sensible al context.

Com a treball futur estem interessats a abordar les següents tasques i problemes:

- Aprofitar la presència del component  $\mathcal{S}$  en la definició de problema sensible al context per a definir nous processadors que tinguin en compte aquest component de forma específica. Sobre aquesta configuració, un altre tema interessant d'investigació és el desenvolupament de tècniques per a generar triples de  $\mu$ -reducció que són utilitzats en el processador de triples de  $\mu$ -reducció.
- Millorar l'estimació del graf de dependència usant les tècniques desenvolupades en [Mid02], on les tècniques basades en autòmats d'arbres s'usen per a obtenir una estimació més precisa del problema derivat de la abastabilitat en la definició de graf de dependència.
- Integrar l'ús del filtrat d'arguments en el processador de triples de  $\mu$ -reducció amb regles usables.
- Obtenir una anàlisi modular de la reescriptura sensible al context usant parells de dependència sensibles al context. La modularitat de la reescriptura sensible al context va ser investigada per primera vegada en [GL02a, GL02b]. El nostre pla és considerar els *mòduls de reescriptura* de Urbain [Urb04] i la seua anàlisi de la modularitat de la terminació per a estendre els seus resultats a la reescriptura sensible al context.
- Provar no-terminació de la reescriptura sensible al context. Recentment, Thiemann i Sternagel [TS09] han investigat els bucles en les computacions sensibles al context. Els seus resultats han sigut utilitzats per a millorar l'última versió de MU-TERM amb un processador per a la detecció de no-terminació de la reescriptura sensible al context més potent (veure [AGLNM10]). Planifiquem usar la nostra experiència en l'anàlisi de les seqüències de  $\mu$ -reescriptura per a aconseguir un processador de no- $\mu$ -terminació més avançat.
- Comparar els *ordres de camins semàntics monòtons* [BFR00] (sensibles al context) i el nostre marc de parells de dependència sensibles al context. És ben conegut que els parells de dependència es poden veure com un cas especial d'ordre de camins semàntics monòtons per a la terminació de

la reescriptura. En la seua tesi doctoral, Borralleras va demostrar que l'ordre de camins semàntics monòtons es pot adaptar per a provar la terminació de la reescriptura sensible al context[Bor03]. Des d'un punt de vista teòric, és interessant explorar com els parells de dependència sensibles al context estan connectats amb l'ordre de camins semàntics monòtons sensible al context de Borralleras.

- Certificar les proves de terminació per als sistemes de reescriptura sensibles al context, que actualment és un tema d'investigació candent. A causa de la seua complexitat i grandària, el desenvolupament d'eines de terminació pot contenir errors d'implementació que sovint comporten errors en les proves de terminació. Per aquesta raó, el desenvolupament d'eines que puguen usar assistents de proves com Coq o Isabelle, que poden validar que l'eixida de l'eina de terminació és una prova correcta, apareix com un complement altament desitjable en les eines de terminació. Pel que fa a la reescriptura sensible al context, encara no s'han desenvolupat biblioteques apropiades ni definicions de Coq/Isabelle de tècniques específiques per a provar la terminació de la reescriptura sensible al context.



---

# Bibliography

- [AAC<sup>+</sup>08] E. Albert, P. Arenas, M. Codish, S. Genaim, G. Puebla, and D. Zanardini. Termination Analysis of Java Bytecode. In G. Barthe and F. S. de Boer, editors, *Proc. of the 10th IFIP WG 6.1 International Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'08*, volume 5051 of *LNCS*, pages 2–18. Springer-Verlag, 2008.
- [AEF<sup>+</sup>08] B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving Context-Sensitive Dependency Pairs. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *LNCS*, pages 636–651. Springer-Verlag, 2008.
- [AEGL10] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. On-Demand Strategy Annotations Revisited: An Improved On-Demand Evaluation Strategy. *Theoretical Computer Science*, 411(2):504–541, 2010.
- [AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science*, 236(1–2):133–178, 2000.
- [AGIL07] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007. Proc. of the 6th Spanish Conference on Programming and Languages, PROLE'06.
- [AGL06] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In S. Arun-Kumar and N. Garg, editors, *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *LNCS*, pages 297–308. Springer-Verlag, 2006.

- [AGL07] B. Alarcón, R. Gutiérrez, and S. Lucas. Improving the Context-Sensitive Dependency Graph. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007. Proc. of the 6th Spanish Conference on Programming and Languages, PROLE'06.
- [AGL10] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. *Information and Computation*, 208:922–968, 2010.
- [AGLNM10] B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. Proving Termination Properties with MU-TERM. In *Proc. of the 13th International Conference on Algebraic Methodology and Software Technology, AMAST'10*, LNCS. Springer-Verlag, 2010. To appear.
- [AL07] B. Alarcón and S. Lucas. Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In F. Wolter, editor, *Proc. of the 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*, volume 4720 of *LNAI*, pages 73–87. Springer-Verlag, 2007.
- [ALNM09] B. Alarcón, S. Lucas, and R. Navarro-Marset. Using Matrix Interpretations over the Reals in Proofs of Termination. In P. Lucio, G. Moreno, and R. Peña, editors, *Proc. of the 9th Spanish Conference on Programming and Languages, PROLE'09*, pages 255–264, 2009.
- [Art96] T. Arts. Termination by Absence of Infinite Chains of Dependency Pairs. In H. Kirchner, editor, *Proc. of the 21st International Colloquium on Trees in Algebra and Programming, CAAP'96*, volume 1059 of *LNCS*, pages 196–210. Springer-Verlag, 1996.
- [Art97] T. Arts. *Automatically Proving Termination and Innermost Normalisation of Term Rewriting Systems*. PhD thesis, Universiteit Utrecht, Utrecht, Netherlands, 1997.
- [Awo06] S. Awodey. *Category Theory*. Clarendon Press, 2006.
- [BFR00] C. Borralleras, M. Ferreira, and A. Rubio. Complete Monotonic Semantic Path Orderings. In D. McAllester, editor, *Proc. of the 17th International Conference on Automated Deduction*,

- CADE'00*, volume 1831 of *LNAI*, pages 346–364. Springer-Verlag, 2000.
- [Bir98] R. Bird. *Introduction to Functional Programming using Haskell*. Pearson Education, 1998.
- [BJM00] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and Proof in Membership Equational Logic. *Theoretical Computer Science*, 236:35–132, 2000.
- [BL87] F. Bellegarde and P. Lescanne. Transformation Ordering. In H. Ehrig, R. Kowalski, G. Levi, and U. Montanari, editors, *Proc. of the International Joint Conference on Theory and Practice of Software Development, TAPSOFT'87*, volume 249 of *LNCS*, pages 69–80. Springer-Verlag, 1987.
- [BLNM<sup>+</sup>09] C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio. Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. In R. A. Schmidt, editor, *Proc. of the 22th Conference on Automated Deduction, CADE'09*, volume 5663 of *LNAI*, pages 294–305. Springer-Verlag, 2009.
- [BLR02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor, *Proc. of the 18th Conference on Automated Deduction, CADE'02*, volume 2392 of *LNAI*, pages 314–331. Springer-Verlag, 2002.
- [BN98] F. Baader and T. Nipkow. *Term Rewriting and All That*. Cambridge University Press, 1998.
- [Bor03] C. Borralleras. *Ordering-Based Methods for Proving Termination Automatically*. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Barcelona, Spain, 2003.
- [CDE<sup>+</sup>07] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and C. Talcott. *All About Maude – A High-Performance Logical Framework*, volume 4350 of *LNCS*. Springer-Verlag, 2007.
- [Chu32] A. Church. A Set of Postulates for the Foundation of Logic. *Annals of Mathematics*, 33(2):346–366, 1932.

- [CL87] A. B. Cherifa and P. Lescanne. Termination of Rewriting Systems by Polynomial Interpretations and its Implementation. *Science of Computer Programming*, 9(2):137–159, 1987.
- [CR36] A. Church and B. Rosser. Some Properties of Conversion. *Transactions of the American Mathematical Society*, 39(3):472–482, 1936.
- [Der79] N. Dershowitz. Orderings for term-rewriting systems. In *Proc. of the 20th Annual Symposium on Foundations of Computer Science, SFCS'79*, pages 123–131. IEEE Computer Society, 1979.
- [Der87] N. Dershowitz. Termination of Rewriting. *Journal of Symbolic Computation*, 3(1-2):69–115, 1987.
- [Der04] N. Dershowitz. Termination by Abstraction. In B. Demoen and V. Lifschitz, editors, *Proc. of the 20th International Conference on Logic Programming, ICLP'04*, volume 3132 of *LNAI*, pages 1–18. Springer-Verlag, 2004.
- [DLM<sup>+</sup>04] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In *Proc. of the 2004 ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147–158. ACM Press, 2004.
- [DLM<sup>+</sup>08] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher Order Symbolic Computation*, 21(1-2):59–88, 2008.
- [EH09] J. Endrullis and D. Hendriks. From Outermost to Context-Sensitive Rewriting. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*, volume 5595 of *LNCS*, pages 305–319. Springer-Verlag, 2009.
- [EH10] J. Endrullis and D. Hendriks. Transforming Outermost into Context-Sensitive Rewriting. *Logical Methods in Computer Science*, 6(2):1–39, 2010.
- [End09] J. Endrullis. Jambox, Automated Termination Proofs For String and Term Rewriting, 2009. Available at <http://joerg.endrullis.de/jambox.html>.

- [EWZ08] J. Endrullis, J. Waldmann, and H. Zantema. Matrix Interpretations for Proving Termination of Term Rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.
- [Fer05] M.-L. Fernández. Relaxing Monotonicity for Innermost Termination. *Information Processing Letters*, 93(3):117–123, 2005.
- [FGJM85] K. Futatsugi, J. A. Goguen, J.-P. Jouannaud, and J. Meseguer. Principles of OBJ2. In *Proc. of the 12th ACM SIGACT-SIGPLAN Symposium on Principles of Programming Languages, POPL'85*, pages 52–66. ACM, 1985.
- [FGK03] O. Fissore, I. Gnaedig, and H. Kirchner. Simplification and Termination of Strategies in Rule-Based Languages. In *Proc. of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP'03*, pages 124–135. ACM, 2003.
- [FK08] S. Falke and D. Kapur. Dependency Pairs for Rewriting with Built-in Numbers and Semantic Data Structures. In A. Voronkov, editor, *Proc. of the 19th International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of *LNCS*, pages 94–109, 2008.
- [FK09] S. Falke and D. Kapur. Termination of Context-Sensitive Rewriting with Built-In Numbers and Collection Data Structures. In S. Escobar, editor, *Proc. of the 18th International Workshop on Functional and (Constraint) Logic Programming, WFLP'09*, pages 111–125, 2009.
- [FKW00] W. Fokkink, J. Kamperman, and P. Walters. Lazy rewriting on eager machinery. *ACM Transactions on Programming Languages and Systems*, 22(1):45–86, 2000.
- [FN97] K. Futatsugi and A. Nakagawa. An overview of CAFE Specification Environment—An algebraic approach for creating, verifying, and maintaining formal specifications over networks. In *Proc. of the 1st International Conference on Formal Engineering Methods, ICFEM '97*, page 170. IEEE Computer Society, 1997.
- [FR99] M. C. F. Ferreira and A. L. Ribeiro. Context-Sensitive AC-Rewriting. In P. Narendran and M. Rusinowitch, editors, *Proc. of the 10th International Conference on Rewriting Techniques*

- and Applications, RTA'99*, volume 1631 of *LNCS*, pages 173–187. Springer-Verlag, 1999.
- [GAO02] J. Giesl, T. Arts, and E. Ohlebusch. Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation*, 34(1):21–58, 2002.
- [GL02a] B. Gramlich and S. Lucas. Modular Termination of Context-Sensitive Rewriting. In *Proc. of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP'02*, pages 50–61. ACM Press, 2002.
- [GL02b] B. Gramlich and S. Lucas. Simple Termination of Context-Sensitive Rewriting. In B. Fischer and E. Visser, editors, *Proc. of the 3rd ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02*, pages 29–41. ACM Press, 2002.
- [GL10a] R. Gutiérrez and S. Lucas. Proving Termination in the Context-Sensitive Dependency Pair Framework. In P. Ölveczky, editor, *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *LNCS*, pages 19–35. Springer-Verlag, 2010.
- [GL10b] R. Gutiérrez and S. Lucas. Proving Termination in the Context-Sensitive Dependency Pairs Framework (Extended Version). Technical report, Universidad Politécnica de Valencia, February 2010. Available as Technical Report DSIC-II/02/10.
- [GLU08] R. Gutiérrez, S. Lucas, and X. Urbain. Usable Rules for Context-Sensitive Rewrite Systems. In A. Voronkov, editor, *Proc. of the 19th International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of *LNCS*, pages 126–141. Springer-Verlag, 2008.
- [GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of the 10th International Conference on Rewriting Techniques and Applications, RTA'99*, volume 1631 of *LNCS*, pages 271–285. Springer-Verlag, 1999.
- [GM04] J. Giesl and A. Middeldorp. Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming*, 14(4):379–427, 2004.

- [Gra94] B. Gramlich. Generalized Sufficient Conditions for Modular Termination of Rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 5:131–151, 1994.
- [GSKT06] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of the 3rd International Joint Conference on Automated Reasoning, IJ-CAR'06*, volume 4130 of *LNAI*, pages 281–286. Springer-Verlag, 2006. Available at <http://www-i2.informatik.rwth-aachen.de/AProVE>.
- [GSSKT06] J. Giesl, P. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated Termination Analysis for Haskell: From Term Rewriting to Programming Languages. In F. Pfenning, editor, *Proc. of the 18th International Conference on Rewriting Techniques and Applications, RTA'06*, volume 4098 of *LNCS*, pages 297–312. Springer-Verlag, 2006.
- [GTSK04] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors, *Proc. of the 11th International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04*, volume 3452 of *LNAI*, pages 301–331. Springer-Verlag, 2004.
- [GTSK05] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In B. Gramlich, editor, *Proc. of the 5th International Workshop on Frontiers of Combining Systems, FroCoS'05*, volume 3717 of *LNAI*, pages 216–231. Springer-Verlag, 2005.
- [GTSKF04] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated Termination Proofs with AProVE. In V. van Oostrom, editor, *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04*, volume 3091 of *LNCS*, pages 210–220. Springer-Verlag, 2004.
- [GTSKF06] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning*, 37(3):155–203, 2006.

- [Gut75] J. V. Guttag. *The Specification and Application to Programming of Abstract Data Types*. PhD thesis, Dept. of Computer Science, University of Toronto, Toronto, Canada, 1975.
- [GWM<sup>+</sup>00] J. A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, and J. P. Jouannaud. *Software Engineering with OBJ: Algebraic Specification in Action*, chapter Introducing OBJ. Kluwer, 2000.
- [Han94] M. Hanus. The Integration of Functions Into Logic Programming. *Journal of Logic Programming*, 19&20:583–628, 1994.
- [HL78] G. Huet and D. S. Lankford. On the Uniform Halting Problem for Term Rewriting Systems. Technical report, Rapport Laboria 283, IRIA, Rocquencourt, France, 1978.
- [HM04] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In V. van Oostrom, editor, *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04*, volume 3091 of *LNCS*, pages 249–268. Springer-Verlag, 2004.
- [HM05] N. Hirokawa and A. Middeldorp. Automating the Dependency Pair Method. *Information and Computation*, 199:172–199, 2005.
- [HM07] N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool: Techniques and Features. *Information and Computation*, 205(4):474–511, 2007. Available at <http://cl-informatik.uibk.ac.at/software/ttt2/>.
- [HPJW92] P. Hudak, S. L. Peyton-Jones, and P. Wadler. Report on the Functional Programming Language Haskell: a non-strict, purely functional language. *Sigplan Notices*, 27(5):1–164, 1992.
- [Hul80] J.-M. Hullot. Canonical Forms and Unification. In W. Bibel and R. Kowalski, editors, *Proc. of the 5th Conference on Automated Deduction, CADE'80*, volume 87 of *LNCS*, pages 318–334. Springer-Verlag, 1980.
- [Itu67] R. Iturriaga. *Contributions to Mechanical Mathematics*. PhD thesis, Dept. of Computer Science, Carnegie-Melon University, Pittsburgh, PA, USA, 1967.
- [Jon03] S. P. Jones, editor. *Haskell 98 Language and Libraries*. Cambridge University Press, 2003.

- [KB70] D. E. Knuth and P. B. Bendix. Simple Word Problems in Universal Algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–267. Pergamon, 1970.
- [KKS91] R. Kennaway, J. W. Klop, M. R. Sleep, and F.-J. de Vries. Transfinite Reductions in Orthogonal Term Rewriting Systems (Extended Abstract). In R. V. Book, editor, *Proc. of the 4th International Conference on Rewriting Techniques and Applications, RTA '91*, LNCS, pages 1–12. Springer-Verlag, 1991.
- [KNS85] D. Kapur, P. Narendran, and G. Sivakumar. A Path Ordering for Proving Termination of Term Rewriting Systems. In H. Ehrig, C. Floyd, M. Nivat, and J. W. Thatcher, editors, *Proc. of the 10th Colloquium on Trees in Algebra and Programming, CAAP'85*, volume 185 of LNCS, pages 173–187. Springer-Verlag, 1985.
- [KNT99] K. Kusakari, M. Nakamura, and Y. Toyama. Argument Filtering Transformation. In G. Nadathur, editor, *Proc. of the International Conference on Principles and Practice of Declarative Programming, PPDP'99*, volume 1702 of LNCS, pages 47–61. Springer-Verlag, 1999.
- [KSZM09] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*, volume 5595 of LNCS, pages 295–304. Springer-Verlag, 2009.
- [Lan79] D. S. Lankford. On Proving Term Rewriting Systems are Noetherian. Technical report, Louisiana Technical University, Ruston, LA, 1979.
- [Les82] P. Lescanne. Some Properties of Decomposition Ordering, a Simplification Ordering to Prove Termination of Rewriting Systems. *RAIRO Informatique théorique*, 16(4):331–347, 1982.
- [LM08] S. Lucas and J. Meseguer. Order-Sorted Dependency Pairs. In S. Antoy and E. Albert, editors, *Proc. of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08*, pages 108–119. ACM Press, 2008.

- [LMM05] S. Lucas, C. Marché, and J. Meseguer. Operational Termination of Conditional Term Rewriting Systems. *Information Processing Letters*, 95(4):446–453, 2005.
- [Luc96] S. Lucas. Termination of Context-Sensitive Rewriting by Rewriting. In F. Meyer auf der Heide and B. Monien, editors, *Proc. of the 23rd International Colloquium on Automata, Languages and Programming, ICALP'96*, volume 1099 of *LNCS*, pages 122–133. Springer-Verlag, 1996.
- [Luc98] S. Lucas. Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, 1998.
- [Luc01] S. Lucas. Termination of On-Demand Rewriting and Termination of OBJ Programs. In R. De Nicola and H. Søndergaard, editors, *Proc. of the 3rd International Conference on Principles and Practice of Declarative Programming, PPDP'01*, pages 82–93. ACM Press, 2001.
- [Luc02] S. Lucas. Context-Sensitive Rewriting Strategies. *Information and Computation*, 178(1):293–343, 2002.
- [Luc04a] S. Lucas. Polynomials for Proving Termination of Context-Sensitive Rewriting. In I. Walukiewicz, editor, *Proc. of the 7th International Conference on Foundations of Software Science and Computation Structures, FOSSACS'04*, volume 2987 of *LNCS*, pages 318–332. Springer-Verlag, 2004.
- [Luc04b] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04*, volume 3091 of *LNCS*, pages 200–209. Springer-Verlag, 2004. Available at <http://zenon.dsic.upv.es/muterm/>.
- [Luc05] S. Lucas. Polynomials over the Reals in Proofs of Termination: from Theory to Practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547–586, 2005.
- [Luc06] S. Lucas. Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation*, 204(12):1782–1846, 2006.

- [McC60] J. McCarthy. Recursive Functions of Symbolic Expressions and their Computation by Machine, Part I. *Communications of the ACM*, 3(4):184–195, 1960.
- [McC63] J. McCarthy. A Basis for a Mathematical Theory of Computation. *Computer Programming and Formal Systems*, pages 33–70, 1963.
- [MG95] A. Middeldorp and B. Gramlich. Simple Termination is Difficult. *Applicable Algebra in Engineering, Communication and Computing*, 6(2):115–128, 1995.
- [Mid97] A. Middeldorp. Call by Need Computations to Root-Stable Form. In *Proc. of the 24th ACM SIGPLAN-SIGACT symposium on Principles of Programming Languages, POPL’97*, pages 94–105. ACM, 1997.
- [Mid02] A. Middeldorp. Approximations for Strategies and Termination. *Electronic Notes in Theoretical Computer Science*, 70(6):1–20, 2002. Proc. of 2nd International Workshop on Reduction Strategies in Rewriting and Programming - Final Proceedings (FLoC Satellite Event), WRS’02.
- [MN70] Z. Manna and S. Ness. On the Termination of Markov Algorithms. In *Proc. of the 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.
- [New42] M. H. A. Newman. On Theories with a Combinatorial Definition of “Equivalence”. *Annals of Mathematics*, 43(2):223–243, 1942.
- [NSEP91] E. G. J. M. H. Nöcker, J. E. W. Smesters, M. C. J. D. van Eekelen, and M. J. Plasmeijer. Concurrent Clean. In E.H.L. Aarts, J. Leeuwen, and M. Rem, editors, *Proc. on Parallel Architectures and Languages Europe, PARLE’91*, LNCS, pages 202–219. Springer-Verlag, 1991.
- [OGS08] B. O’Sullivan, J. Goerzen, and D. Stewart. *Real World Haskell*. O’Reilly Media, Inc., 2008.
- [Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, 2002.
- [Sch24] M. Schönfinkel. Über die Bausteine der mathematischen Logik. *Mathematische Annalen*, 92:305–316, 1924.

- [SG08a] F. Schernhammer and B. Gramlich. On Operational Termination of Deterministic Conditional Term Rewriting Systems. In T. Uustalu, J. Vain, and J. Ernits, editors, *Proc. of the 20th Nordic Workshop on Programming Theory, NWPT'08 (Extended Abstracts)*, pages 84–86, 2008.
- [SG08b] F. Schernhammer and B. Gramlich. Termination of Lazy Rewriting Revisited. *Electronic Notes in Theoretical Computer Science*, 204:35–51, 2008. Proc. of the 7th International Workshop on Reduction Strategies in Rewriting and Programming, WRS'07.
- [SG09] F. Schernhammer and B. Gramlich. VMTL - A Modular Termination Laboratory. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*, volume 5595 of *LNCS*, pages 285–294. Springer-Verlag, 2009. Available at <http://www.logic.at/vmtl/>.
- [SG10] F. Schernhammer and B. Gramlich. Characterizing and Proving Operational Termination of Deterministic Conditional Term Rewriting Systems. *Journal of Logic and Algebraic Programming*, 79(7):659–688, 2010. The 20th Nordic Workshop on Programming Theory, NWPT'08.
- [SKGAR07] P. Schneider-Kamp, J. Giesl, Serebrenik A., and Thiemann R. Automated Termination Analysis for Logic Programs by Term Rewriting. In R. K. Shyamasundar, editor, *Proc. of the 16th International Symposium on Logic-Based Program Synthesis and Transformation, LOPSTR'06*, volume 4407 of *LNCS*, pages 177–193. Springer-Verlag, 2007.
- [ST00] M. Steinby and W. Thomas. Trees and Term Rewriting in 1910: On a Paper by Axel Thue. *EATCS Bull.* 72, pages 256–269, 2000.
- [TeR03] TeReSe. *Term Rewriting Systems*. Cambridge University Press, 2003.
- [TGSK04] R. Thiemann, J. Giesl, and P. Schneider-Kamp. Improved Modular Termination Proofs Using Dependency Pairs. In D. Basin and M. Rusinowitch, editors, *Proc. of the 2nd International Joint Conference on Automated Reasoning, IJCAR'04*, volume 3097 of *LNAI*, pages 75–90. Springer-Verlag, 2004.

- [Thi08] R. Thiemann. *The DP Framework for Proving Termination of Term Rewriting*. PhD thesis, Der Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch Westfälischen Technischen Hochschule Aachen, Aachen, Germany, 2008. Available as Technical Report AIB-2007-17.
- [Thu10] A. Thue. Die Lösung eines Spezialfalles eines generellen logischen Problems. *Kra. Videnskabs-Selskabets Skrifter*, I. Mat. Nat. Kl.(8):273–310, 1910. Reprinted in [Thu77].
- [Thu77] A. Thue. Die Lösung eines Spezialfalles eines generellen logischen Problems. *Selected Mathematical Papers of Axel Thue*, Universitetsforlaget, 1977.
- [Toy87] Y. Toyama. Counterexamples to Termination for the Direct Sum of Term Rewriting Systems. *Information Processing Letters*, 25:141–143, 1987.
- [TS09] R. Thiemann and C. Sternagel. Loops under Strategies. In R. Treinen, editor, *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA '09*, volume 5595 of *LNCS*, pages 17–31. Springer-Verlag, 2009.
- [Urb04] X. Urbain. Modular & Incremental Automated Termination Proofs. *Journal of Automated Reasoning*, 32(4):315–355, 2004.
- [USS08] K. Uchiyama, M. Sakai, and T. Sakabe. Decidability of Innermost Termination and Context-Sensitive Termination for Semi-Constructor Term Rewriting Systems. *Electronic Notes Theoretical Computer Science*, 204:21–34, 2008. Proc. of the 7th International Workshop on Reduction Strategies in Rewriting and Programming, WRS'07.
- [Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of the 7th International Conference on Rewriting Techniques and Applications, RTA '97*, volume 1232 of *LNCS*, pages 172–186. Springer-Verlag, 1997.
- [ZR10] H. Zantema and M. Raffelsieper. Proving Productivity in Infinite Data Structures. In C. Lynch, editor, *Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA '10*, volume 6 of *Leibniz International Proceedings in*

*Informatics (LIPICs)*, pages 401–416. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2010.

---

# Index

- $\mu$ -reduction pair, 36
- $\mu$ -reduction triple, 39
- $\mu$ -narrowing, 21
- abstract reduction system, 17
- arity, 2, 17
- chain of CSDPs, 29
  - minimal, 29
- context-sensitive dependency graph, 38
- context-sensitive dependency pair, 28
- context-sensitive rewriting, 6
- CS problem, 33
  - finite, 33
  - infinite, 33
- CS processor, 34
  - complete, 34
  - sound, 34
- CSDG, 38
- CSDP, 28
- defined, 4
- DP problem, 5
  - finite, 5
  - infinite, 5
- DP processor, 5
  - complete, 5
  - sound, 5
- hidden term, 26
- mgu, 19
- most general unifier, 19
- narrowing, 20
- normal form, 2
- position, 18
  - $\mu$ -replacing, 20
  - active, 20
  - frozen, 20
  - non- $\mu$ -replacing, 20
- reduction orderings, 3
- reduction pair, 36
- relation
  - $\mu$ -monotonic, 20
  - monotonic, 19
  - stable, 19
- rewrite rule, 2, 19
  - collapsing, 19
  - redex, 20
- rules, 2
- signature, 2, 17
- simplification orderings, 4
- subterm, 18
  - strict, 18, 21
- symbol
  - constructor, 20
  - defined, 20
  - hidden, 27
- term
  - ground, 18
  - linear, 18
  - matching, 19
  - minimal non- $\mu$ -terminating, 24
  - minimal nonterminating, 23
  - renaming, 19

- strongly minimal non- $\mu$ -terminating,  
25
- term rewriting, 2
- Termination, 1
- terms, 2
- TRS, 19
  
- unifier, 19

## Part II

# Publications Associated to the Thesis



---

# List of Publications

1. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. In S. Arun-Kumar and N. Garg, editors, *Proc. of XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science*, pages 297–308. Springer-Verlag, 2006.
2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph**. *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.
3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with MU-TERM**. *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.
4. R. Gutiérrez, S. Lucas, and X. Urbain. **Usable Rules for Context-Sensitive Rewrite Systems**. In A. Voronkov, editor, *Proc. of XIX International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of *Lecture Notes in Computer Science*, pages 126–141, Hagenberg, Austria, 2008. Springer-Verlag.
5. B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. **Improving Context-Sensitive Dependency Pairs**. In I. Cervesato, H. Veith, and A. Voronkov, editors, *Proc. of XV International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08*, volume 5330 of *Lecture Notes in Computer Science*, pages 636–651. Springer-Verlag, 2008.
6. B. Alarcón, R. Gutiérrez, and S. Lucas. **Context-Sensitive Dependency Pairs**. *Information and Computation*, 208:922–968, 2010.
7. R. Gutiérrez and S. Lucas. **Proving Termination in the Context-Sensitive Dependency Pairs Framework**. In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2010.
8. B. Alarcón, R. Gutiérrez, S. Lucas, and R. Navarro-Marset. **Proving Termination Properties with MU-TERM**. In M. Johnson and D. Pavlovic, editors, *Proc. of 13th International Conference on Algebraic Methodology And Software Technology, AMAST'10, Lecture Notes in Computer Science*, to appear, 2010. Springer-Verlag.



---

# Publications (full text)

## 8.6 Context-Sensitive Dependency Pairs

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

## Context-Sensitive Dependency Pairs\*

Beatriz Alarcón, Raúl Gutiérrez, and Salvador Lucas

DSIC, Universidad Politécnica de Valencia, Spain  
{balarcon, rgutierrez, slucas}@dsic.upv.es

**Abstract.** Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. Although there is a good number of techniques for proving termination of context-sensitive rewriting (*CSR*), the dependency pair approach, one of the most powerful techniques for proving termination of rewriting, has not been investigated in connection with proofs of termination of *CSR*. In this paper, we show how to use dependency pairs in proofs of termination of *CSR*. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of *CSR*.

**Keywords:** Dependency pairs, term rewriting, program analysis, termination.

### 1 Introduction

A *replacement map* is a mapping  $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$  satisfying  $\mu(f) \subseteq \{1, \dots, k\}$ , for each  $k$ -ary symbol  $f$  of a signature  $\mathcal{F}$  [Luc98]. We use them to discriminate the argument positions on which the rewriting steps are allowed. In this way, for a given Term Rewriting System (TRS [Ohl02, Ter03]), we obtain a restriction of rewriting which we call *context-sensitive rewriting* (*CSR* [Luc98, Luc02]). In *CSR* we only rewrite  $\mu$ -replacing subterms:  $t_i$  is a  $\mu$ -replacing subterm of  $f(t_1, \dots, t_k)$  if  $i \in \mu(f)$ ; every term  $t$  (as a whole) is  $\mu$ -replacing by definition. With *CSR* we can *achieve* a terminating behavior with non-terminating TRSs, by pruning (all) infinite rewrite sequences. Proving termination of *CSR* has been recently recognized as an interesting problem with several applications in the fields of term rewriting and programming languages (see [DLMMU06, GM04, Luc02, Luc06]).

Several methods have been developed for proving termination of *CSR* under a replacement map  $\mu$  for a given TRS  $\mathcal{R}$  (i.e., for proving the  $\mu$ -*termination* of  $\mathcal{R}$ ). In particular, a number of transformations which permit to treat termination of *CSR* as a standard termination problem have been described (see

---

\* This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grant TIN 2004-7943-C04-02, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

[GM04, Luc06] for recent surveys). Direct techniques like polynomial orderings and the context-sensitive version of the recursive path ordering have also been investigated [BLR02, GL02, Luc04b, Luc05]. Up to now, however, the *dependency pairs method* [AG00, GAO02, GTS04, HM04], one of the most powerful techniques for proving termination of rewriting, has not been investigated in connection with proofs of termination of *CSR*. In this paper, we address this problem.

Roughly speaking, given a TRS  $\mathcal{R}$ , the dependency pairs associated to  $\mathcal{R}$  conform a new TRS  $\text{DP}(\mathcal{R})$  which (together with  $\mathcal{R}$ ) determines the so-called *dependency chains* whose finiteness or infiniteness characterize termination of  $\mathcal{R}$ . Given a rewrite rule  $l \rightarrow r$ , we get dependency pairs  $l^\sharp \rightarrow s^\sharp$  for all subterms  $s$  of  $r$  which are rooted by a defined symbol<sup>1</sup>; the notation  $t^\sharp$  for a given term  $t$  means that the root symbol  $f$  of  $t$  is *marked* thus becoming  $f^\sharp$  (often just capitalized:  $F$ ). A chain of dependency pairs is a sequence  $u_i \rightarrow v_i$  of dependency pairs such that  $\sigma(v_i)$  rewrites to  $\sigma(u_{i+1})$  for some substitution  $\sigma$  and  $i \geq 1$ . The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. These basic intuitions are valid for *CSR*, although some important differences arise.

*Example 1.* Consider the following TRS  $\mathcal{R}$  [GM99, Example 1]:

$$\begin{array}{l} c \rightarrow a \quad f(a, b, x) \rightarrow f(x, x, x) \\ c \rightarrow b \end{array}$$

together with  $\mu(f) = \{3\}$ . As shown by Giesl and Middeldorp, among all existing transformations for proving termination of *CSR*, only the *complete* Giesl and Middeldorp's transformation [GM04] (yielding a TRS  $\mathcal{R}_C^\mu$ ) could be used in this case, but no concrete proof of termination for  $\mathcal{R}_C^\mu$  is known yet. Furthermore,  $\mathcal{R}_C^\mu$  has 13 dependency pairs and the dependency graph contains many cycles. In contrast,  $\mathcal{R}$  has only *one* context-sensitive (CS-)dependency pair

$$F(a, b, x) \rightarrow F(x, x, x)$$

and the corresponding dependency graph has *no* cycle (due to the replacement restrictions, since we extend  $\mu$  by  $\mu(F) = \{3\}$ ). As we show below, a direct (and automatic) proof of  $\mu$ -termination of  $\mathcal{R}$  is easy now.

Basically, the subterms in the right-hand sides of the rules which are considered to build the CS-dependency pairs must be  $\mu$ -replacing terms. However, this is not sufficient to obtain a correct approximation. The following example shows the need of a new kind of dependency pairs.

*Example 2.* Consider the following TRS  $\mathcal{R}$ :

$$\begin{array}{l} a \rightarrow c(f(a)) \\ f(c(x)) \rightarrow x \end{array}$$

together with  $\mu(c) = \emptyset$  and  $\mu(f) = \{1\}$ . There is no  $\mu$ -replacing subterm  $s$  in the right-hand sides of the rules which is rooted by a defined symbol. Thus, there is no 'regular' dependency pair. We could wrongly conclude that  $\mathcal{R}$  is  $\mu$ -terminating, which is not true:

<sup>1</sup> A symbol  $f$  is said to be *defined* in a TRS  $\mathcal{R}$  if  $\mathcal{R}$  contains a rule  $f(l_1, \dots, l_k) \rightarrow r$ .

300 B. Alarcón, R. Gutiérrez, and S. Lucas

$$f(\mathbf{a}) \hookrightarrow_{\mu} \underline{f(c(f(\mathbf{a})))} \quad f(\mathbf{a}) \hookrightarrow_{\mu} \dots$$

Indeed, we must add the following dependency pair

$$F(c(X)) \rightarrow X$$

which would not be allowed in Arts and Giesl's approach [AG00] because the right-hand side is a variable.

After some preliminaries in Section 2, Section 3 introduces the general framework to compute and use context-sensitive dependency pairs for proving termination of *CSR*. The introduction of a new kind of dependency pairs (as in Example 2) leads to a new notion of context-sensitive dependency *chain*. We prove the correctness and completeness of the new approach, i.e., our dependency pairs approach fully characterizes termination of *CSR*. We also show how to use term orderings for proving termination of *CSR* by means of the new approach. Furthermore, we are properly extending Arts and Giesl's approach: whenever  $\mu(f) = \{1, \dots, k\}$  for all  $k$ -ary symbols  $f \in \mathcal{F}$ , *CSR* and ordinary rewriting coincide; coherently, our results boil down into the standard results for the dependency pair approach. Section 4 shows how to compute the (estimated) context-sensitive dependency graph and investigates how to use term orderings together with the dependency graph to achieve automatic proofs of termination of *CSR* within the dependency pairs approach. Section 5 adapts Hirokawa and Middeldorp's subterm criterion [HM04] to *CSR*. Section 6 concludes.

## 2 Preliminaries

Throughout the paper,  $\mathcal{X}$  denotes a countable set of variables and  $\mathcal{F}$  denotes a signature, i.e., a set of function symbols  $\{f, g, \dots\}$ , each having a fixed arity given by a mapping  $ar : \mathcal{F} \rightarrow \mathbb{N}$ . The set of terms built from  $\mathcal{F}$  and  $\mathcal{X}$  is  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ . Positions  $p, q, \dots$  are represented by chains of positive natural numbers used to address subterms of  $t$ . Given positions  $p, q$ , we denote their concatenation as  $p.q$ . If  $p$  is a position, and  $Q$  is a set of positions,  $p.Q = \{p.q \mid q \in Q\}$ . We denote the topmost position by  $\Lambda$ . The set of positions of a term  $t$  is  $\mathcal{P}os(t)$ . Positions of non-variable symbols in  $t$  are denoted as  $\mathcal{P}os_{\mathcal{F}}(t)$ , and  $\mathcal{P}os_{\mathcal{X}}(t)$  are the positions of variables. The subterm at position  $p$  of  $t$  is denoted as  $t|_p$  and  $t[s]_p$  is the term  $t$  with the subterm at position  $p$  replaced by  $s$ . We write  $t \triangleright s$  if  $s = t|_p$  for some  $p \in \mathcal{P}os(t)$  and  $t \triangleright s$  if  $t \triangleright s$  and  $t \neq s$ . The symbol labelling the root of  $t$  is denoted as  $root(t)$ . A *context* is a term  $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{X})$  with zero or more 'holes'  $\square$  (a fresh constant symbol).

A rewrite rule is an ordered pair  $(l, r)$ , written  $l \rightarrow r$ , with  $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $l \notin \mathcal{X}$  and  $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$ . The left-hand side (*lhs*) of the rule is  $l$  and  $r$  is the right-hand side (*rhs*). A TRS is a pair  $\mathcal{R} = (\mathcal{F}, R)$  where  $R$  is a set of rewrite rules. Given  $\mathcal{R} = (\mathcal{F}, R)$ , we consider  $\mathcal{F}$  as the disjoint union  $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$  of symbols  $c \in \mathcal{C}$ , called *constructors* and symbols  $f \in \mathcal{D}$ , called *defined functions*, where  $\mathcal{D} = \{root(l) \mid l \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} - \mathcal{D}$ .

*Context-sensitive rewriting.* A mapping  $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$  is a *replacement map* (or  $\mathcal{F}$ -map) if  $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \dots, ar(f)\}$  [Luc98]. Let  $M_{\mathcal{F}}$  be the set of all

$\mathcal{F}$ -maps (or  $M_{\mathcal{R}}$  for the  $\mathcal{F}$ -maps of a TRS  $(\mathcal{F}, R)$ ). A binary relation  $R$  on terms is  $\mu$ -monotonic if  $t R s$  implies  $f(t_1, \dots, t_{i-1}, t, \dots, t_k) R f(t_1, \dots, t_{i-1}, s, \dots, t_k)$  for all  $f \in \mathcal{F}$ ,  $i \in \mu(f)$ , and  $t, s, t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . The set of  $\mu$ -replacing positions  $\mathcal{P}os^\mu(t)$  of  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is:  $\mathcal{P}os^\mu(t) = \{\lambda\}$ , if  $t \in \mathcal{X}$  and  $\mathcal{P}os^\mu(t) = \{\lambda\} \cup \bigcup_{i \in \mu(\text{root}(t))} i \cdot \mathcal{P}os^\mu(t|_i)$ , if  $t \notin \mathcal{X}$ . The set of replacing variables of  $t$  is  $\mathcal{V}ar^\mu(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^\mu(t), t|_p = x\}$ . The  $\mu$ -replacing subterm relation  $\triangleright_\mu$  is given by  $t \triangleright_\mu s$  if there is  $p \in \mathcal{P}os^\mu(t)$  such that  $s = t|_p$ . We write  $t \triangleright_\mu s$  if  $t \triangleright_\mu s$  and  $t \neq s$ . In *context-sensitive rewriting* (CSR [Luc98]), we (only) contract replacing redexes:  $t$   $\mu$ -rewrites to  $s$ , written  $t \hookrightarrow_\mu s$  (or  $t \hookrightarrow_{\mathcal{R}, \mu} s$  and even  $t \hookrightarrow s$ ), if  $t \xrightarrow{p} s$  and  $p \in \mathcal{P}os^\mu(t)$ . A TRS  $\mathcal{R}$  is  $\mu$ -terminating if  $\hookrightarrow_\mu$  is terminating. A term  $t$  is  $\mu$ -terminating if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \hookrightarrow_\mu t_2 \hookrightarrow_\mu \dots \hookrightarrow_\mu t_n \hookrightarrow_\mu \dots$  starting from  $t$ . A pair  $(\mathcal{R}, \mu)$  where  $\mathcal{R}$  is a TRS and  $\mu \in M_{\mathcal{R}}$  is often called a CS-TRS.

*Dependency pairs.* Given a TRS  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  a new TRS  $\text{DP}(\mathcal{R}) = (\mathcal{F}^\#, D(R))$  of *dependency pairs* for  $\mathcal{R}$  is given as follows: if  $f(t_1, \dots, t_m) \rightarrow r \in R$  and  $r = C[g(s_1, \dots, s_n)]$  for some defined symbol  $g \in \mathcal{D}$  and  $s_1, \dots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , then  $f^\#(t_1, \dots, t_m) \rightarrow g^\#(s_1, \dots, s_n) \in D(R)$ , where  $f^\#$  and  $g^\#$  are new fresh symbols (called *tuple symbols*) associated to defined symbols  $f$  and  $g$  respectively [AG00]. Let  $\mathcal{D}^\#$  be the set of tuple symbols associated to symbols in  $\mathcal{D}$  and  $\mathcal{F}^\# = \mathcal{F} \cup \mathcal{D}^\#$ . As usual, for  $t = f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^\#$  to denote the *marked term*  $f^\#(t_1, \dots, t_k)$ . Conversely, given a marked term  $t = f^\#(t_1, \dots, t_k)$ , where  $t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^\natural$  to denote the term  $f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . Given  $T \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ , let  $T^\#$  be the set  $\{t^\# \mid t \in T\}$ .

A reduction pair  $(\succeq, \sqsubset)$  consists of a stable and weakly monotonic quasi-ordering  $\succeq$ , and a stable and well-founded ordering  $\sqsubset$  satisfying either  $\succeq \circ \sqsubset \subseteq \sqsubset$  or  $\sqsubset \circ \succeq \subseteq \sqsubset$ . Note that *monotonicity is not required* for  $\sqsubset$ .

### 3 Context-Sensitive Dependency Pairs

Let  $\mathcal{M}_{\infty, \mu}$  be a set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict  $\mu$ -replacing subterm  $s$  of  $t$  (i.e.,  $t \triangleright_\mu s$ ) is  $\mu$ -terminating. Obviously, if  $t \in \mathcal{M}_{\infty, \mu}$ , then  $\text{root}(t)$  is a defined symbol. The following proposition establishes that, given a minimal non- $\mu$ -terminating term  $t \in \mathcal{M}_{\infty, \mu}$ , there are two ways for an infinite  $\mu$ -rewrite sequence to proceed. The first one is by using ‘visible’ parts of the rules which correspond to  $\mu$ -replacing subterms in the right-hand sides which are rooted by a defined symbol. The second one is by showing up ‘hidden’ non- $\mu$ -terminating subterms which are activated by *migrating* variables in a rule  $l \rightarrow r$ , i.e., variables  $x \in \mathcal{V}ar^\mu(r) - \mathcal{V}ar^\mu(l)$  which are *not*  $\mu$ -replacing in the left-hand side  $l$  but become  $\mu$ -replacing in the right-hand side  $r$ .

**Proposition 1.** *Let  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then for all  $t \in \mathcal{M}_{\infty, \mu}$ , there exist  $l \rightarrow r \in R$ , a substitution  $\sigma$  and a term  $u \in \mathcal{M}_{\infty, \mu}$  such*

302 B. Alarcón, R. Gutiérrez, and S. Lucas

that  $t \xrightarrow{*} \sigma(l) \xrightarrow{A} \sigma(r) \succeq_{\mu} u$  and either (1) there is a  $\mu$ -replacing subterm  $s$  of  $r$  such that  $u = \sigma(s)$ , or (2) there is  $x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$  such that  $\sigma(x) \succeq_{\mu} u$ .

Proposition 1 motivates the following.

**Definition 1.** Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . We define  $DP(\mathcal{R}, \mu) = DP_{\mathcal{F}}(\mathcal{R}, \mu) \cup DP_{\mathcal{X}}(\mathcal{R}, \mu)$  to be the set of context-sensitive dependency pairs (CS-DPs) where:

$$DP_{\mathcal{F}}(\mathcal{R}, \mu) = \{l^{\sharp} \rightarrow s^{\sharp} \mid l \rightarrow r \in R, r \succeq_{\mu} s, \text{root}(s) \in \mathcal{D}, l \not\prec_{\mu} s\}$$

and  $DP_{\mathcal{X}}(\mathcal{R}, \mu) = \{l^{\sharp} \rightarrow x \mid l \rightarrow r \in R, x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)\}$ . We extend  $\mu \in M_{\mathcal{F}}$  into  $\mu^{\sharp} \in M_{\mathcal{F}^{\sharp}}$  by  $\mu^{\sharp}(f) = \mu(f)$  if  $f \in \mathcal{F}$ , and  $\mu^{\sharp}(f^{\sharp}) = \mu(f)$  if  $f \in \mathcal{D}$ .

A rule  $l \rightarrow r$  of a TRS  $\mathcal{R}$  is  $\mu$ -conservative if  $\mathcal{V}ar^{\mu}(r) \subseteq \mathcal{V}ar^{\mu}(l)$ , i.e., it does not contain migrating variables;  $\mathcal{R}$  is  $\mu$ -conservative if all its rules are (see [Luc06]). The following result is immediate from Definition 1.

**Proposition 2.** If  $\mathcal{R}$  is a  $\mu$ -conservative TRS, then  $DP(\mathcal{R}, \mu) = DP_{\mathcal{F}}(\mathcal{R}, \mu)$ .

Therefore, in order to deal with  $\mu$ -conservative TRSs  $\mathcal{R}$  we only need to consider the ‘classical’ dependency pairs in  $DP_{\mathcal{F}}(\mathcal{R}, \mu)$ .

*Example 3.* Consider the TRS  $\mathcal{R}$ :

$$g(x) \rightarrow h(x) \quad h(d) \rightarrow g(c) \quad c \rightarrow d$$

together with  $\mu(g) = \mu(h) = \emptyset$  [Zan97, Example 1].  $DP(\mathcal{R}, \mu)$  is:

$$G(x) \rightarrow H(x) \quad H(d) \rightarrow G(c)$$

with  $\mu^{\sharp}(G) = \mu^{\sharp}(H) = \emptyset$ .

If the TRS  $\mathcal{R}$  contains non- $\mu$ -conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

*Example 4.* Consider the TRS  $\mathcal{R}$  [Zan97, Example 5]:

$$\begin{aligned} \text{if}(\text{true}, X, Y) &\rightarrow X & f(X) &\rightarrow \text{if}(X, c, f(\text{true})) \\ \text{if}(\text{false}, X, Y) &\rightarrow Y \end{aligned}$$

with  $\mu(\text{if}) = \{1, 2\}$ . Then,  $DP(\mathcal{R}, \mu)$  is:

$$F(X) \rightarrow \text{IF}(X, c, f(\text{true})) \quad \text{IF}(\text{false}, X, Y) \rightarrow Y$$

with  $\mu^{\sharp}(F) = \{1\}$  and  $\mu(\text{IF}) = \{1, 2\}$ .

Now we introduce the notion of chain of CS-DPs.

**Definition 2 (Chain of CS-DPs).** Let  $(\mathcal{R}, \mu)$  be a CS-TRS. Given  $\mathcal{P} \subseteq DP(\mathcal{R}, \mu)$ , an  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , for  $i \geq 1$  such that there is a substitution  $\sigma$  satisfying both:

1.  $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
2. if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is  $s_i \in T(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) \succeq_{\mu} s_i$  and  $s_i^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ .

for  $i \geq 1$ . Here, as usual we assume that different occurrences of dependency pairs do not share any variable (renamings are used if necessary).

An  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain with  $u_1 \rightarrow v_1 \in \mathcal{P}$  as heading dependency pair is called minimal if  $\sigma(u_1)^\sharp \in M_{\infty, \mu}$  and all dependency pairs in  $\mathcal{P}$  occur infinitely often.

*Remark 1.* When an  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^\sharp)$ -chain is written for a given substitution  $\sigma$ , we write  $\sigma(u) \hookrightarrow_{\text{DP}(\mathcal{R}, \mu), \mu^\sharp} \sigma(v)$  for steps which use a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  but we rather write  $\sigma(u) \hookrightarrow_{\text{DP}(\mathcal{R}, \mu), \mu^\sharp} s^\sharp$  for steps which use a dependency pair  $u \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , where  $s$  is as in Definition 2.

In the following, we use  $\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$  to denote the subset of dependency pairs in  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  whose migrating variables occur on non- $\mu$ -replacing immediate subterms in the left-hand side:

$$\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu) = \{f^\sharp(u_1, \dots, u_k) \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \mid \exists i, 1 \leq i \leq k, i \notin \mu(f^\sharp), x \in \text{Var}(u_i)\}$$

For instance,  $\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  for the CS-TRS  $(\mathcal{R}, \mu)$  in Example 4. For this subset of CS-dependency pairs, we have the following.

**Proposition 3.** *There is no infinite  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain with  $\mathcal{P} \subseteq \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$ .*

The following result establishes the correctness of the context-sensitive dependency pairs approach.

**Theorem 1 (Correctness).** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If there is no infinite  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^\sharp)$ -chain, then  $\mathcal{R}$  is  $\mu$ -terminating.*

As an immediate consequence of Theorem 1 and Proposition 3, we have the following.

**Corollary 1.** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$ , then  $\mathcal{R}$  is  $\mu$ -terminating.*

*Example 5.* Consider the following TRS  $\mathcal{R}$  [Luc98, Example 15]

```

and(true,X) -> X          first(0,X) -> nil
and(false,Y) -> false    first(s(X),cons(Y,Z)) -> cons(Y,first(X,Z))
if(true,X,Y) -> X        from(X) -> cons(X,from(s(X)))
if(false,X,Y) -> Y
add(0,X) -> X
add(s(X),Y) -> s(add(X,Y))

```

with  $\mu(\text{cons}) = \mu(\text{s}) = \mu(\text{from}) = \emptyset$ ,  $\mu(\text{add}) = \mu(\text{and}) = \mu(\text{if}) = \{1\}$ , and  $\mu(\text{first}) = \{1, 2\}$ . Then,  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$  is:

```

ADD(0,X) -> X          IF(true,X,Y) -> X
AND(true,X) -> X      IF(false,X,Y) -> Y

```

Thus, by Corollary 1 we conclude the  $\mu$ -termination of  $\mathcal{R}$ .

Now we prove that the previous CS-dependency pairs approach is not only correct but also complete for proving termination of CSR.

304 B. Alarcón, R. Gutiérrez, and S. Lucas

**Theorem 2 (Completeness).** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If  $\mathcal{R}$  is  $\mu$ -terminating, then there is no infinite  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^\sharp)$ -chain.*

**Corollary 2 (Characterization of  $\mu$ -termination).** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ .  $\mathcal{R}$  is  $\mu$ -terminating iff there is no infinite  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^\sharp)$ -chain.*

In the dependency pairs approach, the absence of infinite chains is checked by finding a *reduction pair*  $(\succeq, \sqsupset)$  which is compatible with the rules and the dependency pairs [AG00]. In our setting, we can relax the monotonicity requirements and use  $\mu$ -reduction pairs  $(\succ, \sqsupset)$  where  $\succ$  is a stable and  $\mu$ -monotonic quasi-ordering which is compatible with the well-founded and stable ordering  $\sqsupset$ , i.e.,  $\succ \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succ \subseteq \sqsupset$ . The following result shows how to use  $\mu$ -reduction pairs for proving  $\mu$ -termination. This is the context-sensitive counterpart of [AG00, Theorem 7]; however, a number of remarkable differences arise due to the treatment of the dependency pairs in  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . Basically, we need to ensure that the quasi-ordering is able to ‘look’ for a  $\mu$ -replacing subterm inside the instantiation  $\sigma(x)$  of a migrating variable  $x$  (hence we require  $\triangleright_{\mu} \subseteq \succ$ ) and also connect a term which is rooted by defined symbol  $f$  and the corresponding dependency pair which is rooted by  $f^\sharp$  (hence the requirement  $f(x_1, \dots, x_k) \succ f^\sharp(x_1, \dots, x_k)$ ).

**Theorem 3.** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if there is a  $\mu$ -reduction pair  $(\succ, \sqsupset)$  such that,*

1.  $l \succ r$  for all  $l \rightarrow r \in R$ ,
2.  $u \sqsupset v$  for all  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
3. whenever  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \neq \emptyset$  we have that  $\triangleright_{\mu} \subseteq \succ$ , where  $\triangleright_{\mu}$  is the  $\mu$ -replacing subterm relation on  $T(\mathcal{F}, \mathcal{X})$ , and
  - (a)  $u (\succ \cup \sqsupset) v$  for all  $u \rightarrow v \in \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$ ,  $u \sqsupset v$  for all  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) - \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$ , and  $f(x_1, \dots, x_k) \succ f^\sharp(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$ , or
  - (b)  $u (\succ \cup \sqsupset) v$  for all  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  and  $f(x_1, \dots, x_k) \sqsupset f^\sharp(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$ .

## 4 Context-Sensitive Dependency Graph

As noticed by Arts and Giesl, the analysis of infinite sequences of dependency pairs can be made by looking at (the cycles  $\mathfrak{C}$  of) the *dependency graph* associated to the TRS  $\mathcal{R}$ . The nodes of the dependency graph are the dependency pairs in  $\text{DP}(\mathcal{R})$ ; there is an arc from a dependency pair  $u \rightarrow v$  to a dependency pair  $u' \rightarrow v'$  if there are substitutions  $\sigma$  and  $\theta$  such that  $\sigma(v) \rightarrow_{\mathcal{R}}^* \theta(u')$ .

Similarly, in the *context-sensitive (CS-)dependency graph*:

1. There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  to a dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if there are substitutions  $\sigma$  and  $\theta$  such that  $\sigma(v) \hookrightarrow_{\mathcal{R}, \mu^\sharp}^* \theta(u')$ .
2. There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to *each* dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$ .

Note that the use of  $\mu^\sharp$  (which restricts reductions on the arguments of the dependency pair symbols  $f^\sharp$ ) is essential: given a set of dependency pairs associated to a CS-TRS  $(\mathcal{R}, \mu)$ , we have less arcs between them due to the presence of such replacement restrictions.

*Example 6.* Consider the CS-TRS in Example 1.  $\text{DP}(\mathcal{R}, \mu)$  is:

$$F(\mathbf{a}, \mathbf{b}, \mathbf{X}) \rightarrow F(\mathbf{X}, \mathbf{X}, \mathbf{X})$$

with  $\mu^\sharp(F) = \{3\}$ . Although the dependency graph contains a cycle (due to  $\sigma(F(\mathbf{X}, \mathbf{X}, \mathbf{X})) \rightarrow^* \sigma(F(\mathbf{a}, \mathbf{b}, \mathbf{Y}))$  for  $\sigma(X) = \sigma(Y) = \mathbf{c}$ ), the CS-dependency graph contains *no* cycle because it is *not* possible to  $\mu^\sharp$ -reduce  $\theta(F(\mathbf{X}, \mathbf{X}, \mathbf{X}))$  into  $\theta(F(\mathbf{a}, \mathbf{b}, \mathbf{Y}))$  for any substitution  $\theta$  (due to  $\mu^\sharp(F) = \{3\}$ ).

As noticed by Arts and Giesl, the presence of an infinite chain of dependency pairs correspond to a cycle in the dependency graph (but not vice-versa).

Again, as an immediate consequence of Theorem 1 and Proposition 3, we have the following.

**Corollary 3.** *Let  $\mathcal{R}$  be a TRS,  $\mu \in M_{\mathcal{R}}$  and  $\mathfrak{C} \subseteq \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$  be a cycle. Then, there is no minimal  $(\mathcal{R}, \mathfrak{C}, \mu^\sharp)$ -chain.*

According to this, and continuing Example 6, we conclude the  $\mu$ -termination of  $\mathcal{R}$  in Example 1.

#### 4.1 Estimating the CS-Dependency Graph

In general, the (context-sensitive) dependency graph of a TRS is *not* computable and we need to use some approximation of it. Following [AG00], we describe how to approximate the CS-dependency graph of a CS-TRS  $(\mathcal{R}, \mu)$ . Let  $\text{CAP}^\mu$  be given as follows: let  $D$  be a set of defined symbols (in our context,  $D = \mathcal{D} \cup \mathcal{D}^\sharp$ ):

$$\text{CAP}^\mu(x) = x \text{ if } x \text{ is a variable}$$

$$\text{CAP}^\mu(f(t_1, \dots, t_k)) = \begin{cases} y & \text{if } f \in D \\ f([t_1]_1^f, \dots, [t_k]_1^f) & \text{otherwise} \end{cases}$$

where  $y$  is intended to be a new, fresh variable which has not yet been used and given a term  $s$ ,  $[s]_i^f = \text{CAP}^\mu(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ . Let  $\text{REN}^\mu$  given by:  $\text{REN}^\mu(x) = y$  if  $x$  is a variable and  $\text{REN}^\mu(f(t_1, \dots, t_k)) = f([t_1]_1^f, \dots, [t_k]_1^f)$  for every  $k$ -ary symbol  $f$ , where given a term  $s \in \mathcal{T}^\sharp(\mathcal{F}, \mathcal{X})$ ,  $[s]_i^f = \text{REN}^\mu(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ . Then, we have an arc from  $u_i \rightarrow v_i$  to  $u_j \rightarrow v_j$  if  $\text{REN}^\mu(\text{CAP}^\mu(v_i))$  and  $u_j$  unify; following [AG00], we say that  $v_i$  and  $u_j$  are  $\mu$ -connectable. The following result whose proof is similar to that of [AG00, Theorem 21] (we only need to take into account the replacement restrictions indicated by the replacement map  $\mu$ ) formalizes the correctness of this approach.

**Proposition 4.** *Let  $(\mathcal{R}, \mu)$  be a CS-TRS. If there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$  in the CS-dependency graph, then  $v$  and  $u'$  are  $\mu$ -connectable.*

306 B. Alarcón, R. Gutiérrez, and S. Lucas

*Example 7.* (Continuing Ex. 6) Since  $\text{REN}^{\mu^\sharp}(\text{CAP}^{\mu^\sharp}(\mathbf{F}(\mathbf{X}, \mathbf{X}, \mathbf{X}))) = \mathbf{F}(\mathbf{X}, \mathbf{X}, \mathbf{Z})$  and  $\mathbf{F}(\mathbf{a}, \mathbf{b}, \mathbf{Y})$  do not unify, we conclude (and this can be easily implemented) that the CS-dependency graph for the CS-TRS  $(\mathcal{R}, \mu)$  in Example 1 has no cycle.

#### 4.2 Checking $\mu$ -Termination with the Dependency Graph

For the cycles in the dependency graph, the absence of infinite chains is checked by finding (possibly different) *reduction pairs*  $(\succeq_{\mathfrak{C}}, \sqsupset_{\mathfrak{C}})$  for each cycle  $\mathfrak{C}$  [GAO02, Theorem 3.5]. In our setting, we use  $\mu$ -reduction pairs.

**Theorem 4 (Use of the CS-dependency graph).** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if for each cycle  $\mathfrak{C}$  in the context-sensitive dependency graph there is a  $\mu$ -reduction pair  $(\succeq_{\mathfrak{C}}, \sqsupset_{\mathfrak{C}})$  such that,  $R \subseteq \succeq_{\mathfrak{C}}$ ,  $\mathfrak{C} \subseteq \succeq_{\mathfrak{C}} \cup \sqsupset_{\mathfrak{C}}$ , and*

1. *If  $\mathfrak{C} \cap \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$ , then  $\mathfrak{C} \cap \sqsupset_{\mathfrak{C}} \neq \emptyset$*
2. *If  $\mathfrak{C} \cap \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \neq \emptyset$ , then  $\supseteq_{\mu} \subseteq \succeq_{\mathfrak{C}}$  (where  $\supseteq_{\mu}$  is the  $\mu$ -replacing subterm relation on  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ ), and*
  - (a)  *$\mathfrak{C} \cap \sqsupset_{\mathfrak{C}} \neq \emptyset$  and  $f(x_1, \dots, x_k) \succeq_{\mathfrak{C}} f^\sharp(x_1, \dots, x_k)$  for all  $f^\sharp$  in  $\mathfrak{C}$ , or*
  - (b)  *$f(x_1, \dots, x_k) \sqsupset_{\mathfrak{C}} f^\sharp(x_1, \dots, x_k)$  for all  $f^\sharp$  in  $\mathfrak{C}$ .*

Following Hirokawa and Middeldorp, the practical use of Theorem 4 concerns the so-called *strongly connected components* (SCCs) of the dependency graph, rather than the cycles themselves (which are exponentially many) [HM04, HM05]. A strongly connected component in the (CS-)dependency graph is a *maximal cycle*, i.e., it is not contained in any other cycle. According to Hirokawa and Middeldorp, when considering an SCC  $\mathfrak{C}$ , we *remove* from  $\mathfrak{C}$  those pairs  $u \rightarrow v$  satisfying  $u \sqsupset v$ . Then, we recompute the SCCs with the remaining pairs in the CS-dependency graph and start again (see [HM05, Section 4]). In our setting, it is not difficult to see that, if the condition  $f(x_1, \dots, x_k) \sqsupset_{\mathfrak{C}} f^\sharp(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$  holds for a given cycle  $\mathfrak{C}$ , then we can remove from  $\mathfrak{C}$  all dependency pairs in  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , thus continuing from  $\mathfrak{C} - \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ .

*Example 8.* Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 4 and  $\text{DP}(\mathcal{R}, \mu)$ :

```
F(X) -> IF(X, c, f(true))
IF(false, X, Y) -> Y
```

with  $\mu^\sharp(\mathbf{F}) = \{1\}$  and  $\mu^\sharp(\text{IF}) = \{1, 2\}$ . These two CS-dependency pairs form the only cycle in the CS-dependency graph. The  $\mu$ -reduction pair  $(\succeq, >)$  induced by the polynomial interpretation

$$\begin{array}{lll} [c] = [\text{true}] = 0 & [f](x) = x & [F](x) = x \\ [\text{false}] = 1 & [\text{if}](x, y, z) = x + y + z & [\text{IF}](x, y, z) = x + z \end{array}$$

can be used to prove the  $\mu$ -termination of  $\mathcal{R}$ .

The use of *argument filterings*, which is standard in the current formulations of the dependency pairs method, also adapts without changes to the context-sensitive setting. This is a simple consequence of [AG00, Theorem 11] (using  $\mu$ -monotonicity instead of monotonicity for the quasi-orderings is not a problem).

## 5 Subterm Criterion

In [HM04], Hirokawa and Middeldorp introduce a very interesting *subterm criterion* which permits to ignore certain cycles of the dependency graph.

**Definition 3.** [HM04] *Let  $\mathcal{R}$  be a TRS and  $\mathfrak{C} \subseteq \text{DP}(\mathcal{R})$  such that every dependency pair symbol in  $\mathfrak{C}$  has positive arity. A simple projection for  $\mathfrak{C}$  is a mapping  $\pi$  that assigns to every  $k$ -ary dependency pair symbol  $f^\sharp$  in  $\mathfrak{C}$  an argument position  $i \in \{1, \dots, k\}$ . The mapping that assigns to every term  $f^\sharp(t_1, \dots, t_k) \in \mathcal{T}^\sharp(\mathcal{F}, \mathcal{X})$  with  $f^\sharp$  a dependency pair symbol in  $\mathcal{R}$  its argument position  $\pi(f^\sharp)$  is also denoted by  $\pi$ .*

In the following result, for a simple projection  $\pi$  and  $\mathfrak{C} \subseteq \text{DP}(\mathcal{R}, \mu)$ , we let  $\pi(\mathfrak{C}) = \{\pi(u) \rightarrow \pi(v) \mid u \rightarrow v \in \mathfrak{C}\}$ . Note that  $u, v \in \mathcal{T}^\sharp(\mathcal{F}, \mathcal{X})$ , but  $\pi(u), \pi(v) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ .

**Theorem 5.** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\mathfrak{C} \subseteq \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  be a cycle. If there exists a simple projection  $\pi$  for  $\mathfrak{C}$  such that  $\pi(\mathfrak{C}) \subseteq \succeq_\mu$ , and  $\pi(\mathfrak{C}) \cap \triangleright_\mu \neq \emptyset$ , then there is no minimal  $(\mathcal{R}, \mathfrak{C}, \mu^\sharp)$ -chain.*

Note that the result is restricted to cycles which do *not* include dependency pairs in  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . The following result provides a kind of generalization of the subterm criterion to simple projections which only consider *non- $\mu$ -replacing* arguments of tuple symbols.

**Theorem 6.** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\mathfrak{C} \subseteq \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  be a cycle. Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded and  $\pi$  be a simple projection for  $\mathfrak{C}$  such that for all  $f^\sharp$  in  $\mathfrak{C}$ ,  $\pi(f^\sharp) \notin \mu^\sharp(f^\sharp)$  and  $\pi(\mathfrak{C}) \subseteq \succsim$ .*

1. *If  $\mathfrak{C} \cap \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$  and  $\mathfrak{C} \cap > \neq \emptyset$ , then there is no minimal  $(\mathcal{R}, \mathfrak{C}, \mu^\sharp)$ -chain.*
2. *If  $\mathfrak{C} \cap \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) \neq \emptyset$ ,  $\succeq_\mu \subseteq \succsim$  (where  $\succeq_\mu$  is the  $\mu$ -replacing subterm relation on  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ ), and*
  - (a)  *$\mathfrak{C} \cap > \neq \emptyset$  and  $f(x_1, \dots, x_k) \succsim x_{\pi(f^\sharp)}$  for all  $f \in \mathcal{D}$  such that  $f^\sharp$  is in  $\mathfrak{C}$ ,*  
or
  - (b)  *$f(x_1, \dots, x_k) > x_{\pi(f^\sharp)}$  for all  $f \in \mathcal{D}$  such that  $f^\sharp$  is in  $\mathfrak{C}$ ,*  
then there is no minimal  $(\mathcal{R}, \mathfrak{C}, \mu^\sharp)$ -chain.

*Example 9.* Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 3.  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\begin{array}{l} \mathbf{G}(\mathbf{X}) \rightarrow \mathbf{H}(\mathbf{X}) \\ \mathbf{H}(\mathbf{d}) \rightarrow \mathbf{G}(\mathbf{c}) \end{array}$$

where  $\mu^\sharp(\mathbf{G}) = \mu^\sharp(\mathbf{H}) = \emptyset$ . The dependency graph contains a single cycle including both of them. The only simple projection is  $\pi(\mathbf{G}) = \pi(\mathbf{H}) = 1$ . Since  $\pi(\mathbf{G}(\mathbf{X})) = \pi(\mathbf{H}(\mathbf{X}))$ , we only need to guarantee that  $\pi(\mathbf{H}(\mathbf{d})) = \mathbf{d} > \mathbf{c} = \pi(\mathbf{G}(\mathbf{c}))$  holds for a stable and well-founded ordering  $>$ . This is easily fulfilled by, e.g., a polynomial ordering.

308 B. Alarcón, R. Gutiérrez, and S. Lucas

## 6 Conclusions

We have shown how to use dependency pairs in proofs of termination of *CSR*. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of *CSR*. Some interesting differences arise which can be summarized as follows: in sharp contrast to the standard dependency pairs approach, where all dependency pairs have tuple symbols  $f^\sharp$  both in the left- and right-hand sides, we have dependency pairs having a single *variable* in the right-hand side. These variables reflect the effect of the *migrating* variables into the termination behavior of *CSR*. This leads to a new definition of chain of context-sensitive dependency pairs which also differs from the standard approach in that we have to especially deal with such migrating variables. As in Arts and Giesl's approach, the presence or absence of infinite chains of dependency pairs from  $\text{DP}(\mathcal{R}, \mu)$  characterizes the  $\mu$ -termination of  $\mathcal{R}$  (Theorems 1 and 2). Furthermore, we are also able to use term orderings to ensure the absence of infinite chains of context-sensitive dependency pairs (Theorem 3). In fact, we are properly extending Arts and Giesl's approach: whenever  $\mu(f) = \{1, \dots, k\}$  for all  $k$ -ary symbols  $f \in \mathcal{F}$ , *CSR* and ordinary rewriting coincide and all these results and techniques boil down into well-known results and techniques for the dependency pairs approach.

Regarding the practical use of the CS-dependency pairs in proofs of termination of *CSR*, we have shown how to build and use the corresponding CS-dependency graph to either prove that the rules of the TRS and the cycles in the CS-dependency graph are compatible with some reduction pair (Theorem 4) or to prove that there are cycles which do not need to be considered at all (Theorems 5 and 6). We have implemented these ideas as part of the termination tool MU-TERM [AGIL07, Luc04a]. We refer the reader to [AGIL07] for details about the practical impact of the techniques developed in this paper. From this preliminary results, we can well conclude that the CS-dependency pairs can play in *CSR* the (practical and theoretical) role than dependency pairs play in rewriting.

There are many other aspects of the dependency pairs approach which are also worth to be considered and eventually extended to *CSR* (e.g., narrowing refinements, modularity issues, innermost computations, usable rules, ...). These aspects provide an interesting subject for future work.

## References

- [AG00] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
- [AGIL07] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, to appear, 2007.
- [BLR02] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In *Proc. of CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.

- [DLMMU06] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2006.
- [GAO02] J. Giesl, T. Arts, and E. Ohlebusch. Modular Termination Proofs for Rewriting Using Dependency Pairs. *Journal of Symbolic Computation* 34(1):21-58, 2002.
- [GL02] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In *Proc. of RULE'02*, pages 29-41, ACM Press, New York, 2002.
- [GM99] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In *Proc. of RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.
- [GM04] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379-427, 2004.
- [GTS04] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In *Proc. of LPAR'04*, LNCS 3452:301-331, Springer-Verlag, Berlin, 2004.
- [HM04] N. Hirokawa and A. Middeldorp. Dependency Pairs Revisited. In *Proc. of RTA'04*, LNCS 3091:249-268, Springer-Verlag, Berlin, 2004.
- [HM05] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199:172-199, 2005.
- [Luc98] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.
- [Luc02] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.
- [Luc04a] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In *Proc. of RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at <http://www.dsic.upv.es/~slucas/csr/termination/muterm>.
- [Luc04b] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In *Proc. of FOSSACS'04*, LNCS 2987:318-332, Springer-Verlag, Berlin 2004.
- [Luc05] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547-586, 2005.
- [Luc06] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, to appear 2006.
- [Ohl02] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.
- [Ter03] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.
- [Zan97] H. Zantema. Termination of Context-Sensitive Rewriting. In *Proc. of RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.

## 8.7 Improving the Context-Sensitive Dependency Graph

2. B. Alarcón, R. Gutiérrez, and S. Lucas. **Improving the Context-Sensitive Dependency Graph.** *Electronic Notes in Theoretical Computer Science*, 188:91–103, 2007.



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

Electronic Notes in Theoretical Computer Science 188 (2007) 91–103

Electronic Notes in  
Theoretical Computer  
Science[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

# Improving the Context-sensitive Dependency Graph

Beatriz Alarcón, Raúl Gutiérrez, and Salvador Lucas<sup>1,2</sup>*Departamento de Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Valencia, Spain*

---

## Abstract

The dependency pairs method is one of the most powerful technique for proving termination of rewriting and it is currently central in most automatic termination provers. Recently, it has been adapted to be used in proofs of termination of context-sensitive rewriting. The use of *collapsing* dependency pairs i.e., *having a single variable in the right-hand side* is a novel and essential feature to obtain a correct framework in this setting. Unfortunately, dependency pairs behave as a kind of *glue* in the context-sensitive dependency graph which makes the cycles bigger, thus making some proofs of termination harder. In this paper we show that this effect can be safely mitigated by removing some arcs from the graph, thus leading to faster and easier proofs. Narrowing dependency pairs is also introduced and used here to eventually simplify the treatment of the context-sensitive dependency graph. We show the practicality of the new techniques with some benchmarks.

*Keywords:* Dependency pairs, term rewriting, program analysis, termination.

---

## 1 Introduction

Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. With *context-sensitive rewriting* (*CSR* [10,11]) we can *achieve* a terminating behavior with non-terminating Term Rewriting Systems (TRSs [14,15]), by pruning (all) infinite rewrite sequences. In *CSR* we only rewrite  $\mu$ -replacing subterms. Here,  $\mu$  is a *replacement map*, i.e., a mapping  $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$  satisfying  $\mu(f) \subseteq \{1, \dots, k\}$ , for each  $k$ -ary symbol  $f$  of the signature  $\mathcal{F}$  [10]. We use them to discriminate the argument positions on which the rewriting steps are allowed. Then,  $t_i$  is a  $\mu$ -replacing subterm of  $f(t_1, \dots, t_k)$  if  $i \in \mu(f)$ ; every term  $t$  (as a whole) is  $\mu$ -replacing by definition. For other subterms we proceed inductively

---

<sup>1</sup> This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-0007, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

<sup>2</sup> Email: [balarcon,rgutierrez,slucas}@dsic.upv.es](mailto:{balarcon,rgutierrez,slucas}@dsic.upv.es)

in this way. Then, for a given TRS, we obtain a restriction of rewriting which we call *context-sensitive rewriting*. Proving termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and programming languages (see [5,6,8,11,13] for further motivation).

The *dependency pairs method* [1] is one of the most powerful techniques for proving termination of rewriting. Roughly speaking, given a TRS  $\mathcal{R}$ , the dependency pairs associated to  $\mathcal{R}$  conform a new TRS  $DP(\mathcal{R})$  which (together with  $\mathcal{R}$ ) determines the so-called *dependency chains* whose finiteness characterizes termination of  $\mathcal{R}$ . The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. In [3], the dependency pairs method has been adapted to be used in proofs of termination of *CSR*. The technique has been implemented in the tool MU-TERM [2,12]. Basically, the non-variable subterms in the right-hand sides of the rules which are considered to build the CS-dependency pairs must be  $\mu$ -replacing terms. Nevertheless such ‘standard’ dependency pairs do not suffice to obtain a correct method for proving termination of *CSR*.

**Example 1.1** [3, Example 2] Consider the following TRS  $\mathcal{R}$ :

$$\begin{array}{l} a \rightarrow c(\mathbf{f}(a)) \\ \mathbf{f}(c(X)) \rightarrow X \end{array}$$

together with  $\mu(c) = \emptyset$  and  $\mu(\mathbf{f}) = \{1\}$ . There is no  $\mu$ -replacing subterm  $s$  in the right-hand sides of the rules which is rooted by a defined symbol. Thus, there is no ‘standard’ dependency pair. We could wrongly conclude that  $\mathcal{R}$  is  $\mu$ -terminating, which is not true:

$$\mathbf{f}(a) \hookrightarrow_{\mu} \mathbf{f}(c(\mathbf{f}(a))) \hookrightarrow_{\mu} \mathbf{f}(a) \hookrightarrow_{\mu} \dots$$

Indeed, as shown in [3], we must add the following dependency pair

$$\mathbf{F}(c(X)) \rightarrow X$$

which would not be allowed in Arts and Giesl’s approach [1] because the right-hand side is a variable. In this paper, we call *collapsing* to such kind of dependency pairs.

As in Arts and Giesl’s approach, the analysis of infinite sequences of context-sensitive dependency pairs can be made by looking at (the cycles  $\mathcal{C}$  of) the *context-sensitive dependency graph* associated to the CS-TRS  $\mathcal{R}$ . The nodes of the dependency graph are the dependency pairs in  $DP(\mathcal{R}, \mu)$ . A disappointing aspect of *collapsing* context-sensitive dependency pairs (as  $\mathbf{F}(c(X)) \rightarrow X$  above) is that they are connected to *every other* dependency pair in the context-sensitive dependency graph [3]. Intuitively, this is because the variable  $X$  in the right-hand side of the dependency pair could be instantiated to anything, thus being potentially able to ‘connect’ to every other dependency pair.

In this paper, we show that we can restrict the number of outgoing links of collapsing dependency pairs to dependency pairs headed by the so-called *hidden* symbols which are defined symbols that occur in non-replacing positions in the right-hand sides of some rule in the TRS. This leads to a new definition of the context-sensitive dependency graph which greatly improves the performance of the original method.

**Example 1.2** Consider the following non-terminating TRS  $\mathcal{R}$  which can be used to compute the list of prime numbers [7] :

```

primes -> sieve(from(s(s(0))))      tail(cons(X,Y)) -> Y
from(X) -> cons(X,from(s(X)))      if(true,X,Y) -> X
head(cons(X,Y)) -> X               if(false,X,Y) -> Y
filter(s(s(X)),cons(Y,Z)) ->
  if(divides(s(s(X)),Y),filter(s(s(X)),Z),cons(Y,filter(X,sieve(Y))))
sieve(cons(X,Y)) -> cons(X,filter(X,sieve(Y)))

```

together with  $\mu(\text{cons}) = \mu(\text{if}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{ar}(f)\}$  for any other symbols  $f$ . No (automatic or manual) proof of termination for this CS-TRS has been reported to date. By using the dependency graph as defined in [3] we were not able to find a proof with MU-TERM 4.3 [2].

In contrast, with the new definition in this paper, we have *no* cycles! Thus, a direct (and automatic) proof of  $\mu$ -termination of  $\mathcal{R}$  is easy now.

Narrowing dependency pairs was also introduced by Arts and Giesl to improve the efficiency of the dependency pairs technique in proofs of termination [1]. Roughly speaking, under some conditions, a dependency pair can be replaced by a set of pairs which could simplify or restructure the dependency graph and eventually simplify the proof of termination. We also investigate this technique for dealing with the context-sensitive dependency graph.

After some preliminary definitions in Section 2, Section 3 introduces the notion of hidden symbol and investigates its properties in proofs of termination of *CSR*. Section 4 shows how to use it to improve the context-sensitive dependency graph. Section 5 adapts narrowing of dependency pairs to context-sensitive dependency pairs. Section 6 provides an experimental evaluation of our techniques. Section 7 concludes.

## 2 Preliminaries

Throughout the paper,  $\mathcal{X}$  denotes a countable set of variables and  $\mathcal{F}$  denotes a signature, i.e., a set of function symbols  $\{f, g, \dots\}$ , each having a fixed arity given by a mapping  $\text{ar} : \mathcal{F} \rightarrow \mathbb{N}$ . The set of terms built from  $\mathcal{F}$  and  $\mathcal{X}$  is  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ . Positions  $p, q, \dots$  are represented by chains of positive natural numbers used to address subterms of  $t$ . Given positions  $p, q$ , we denote its concatenation as  $p.q$ . If  $p$  is a position, and  $Q$  is a set of positions,  $p.Q = \{p.q \mid q \in Q\}$ . We denote the empty chain by  $\Lambda$ . The set of positions of a term  $t$  is  $\text{Pos}(t)$ . 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  $t \succeq s$  if  $s = t|_p$  for some  $p \in \text{Pos}(t)$  and  $t \triangleright s$  if  $t \succeq s$  and  $t \neq s$ . The symbol labelling the root of  $t$  is denoted as  $\text{root}(t)$ . A *context* is a term  $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{X})$  with zero or more ‘holes’  $\square$  (a fresh constant symbol).

A rewrite rule is an ordered pair  $(l, r)$ , written  $l \rightarrow r$ , with  $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $l \notin \mathcal{X}$  and  $\text{Var}(r) \subseteq \text{Var}(l)$ . The left-hand side (*lhs*) of the rule is  $l$  and  $r$  is the right-hand side (*rhs*). A TRS is a pair  $\mathcal{R} = (\mathcal{F}, R)$  where  $R$  is a set of rewrite rules. Given  $\mathcal{R} = (\mathcal{F}, R)$ , we consider  $\mathcal{F}$  as the disjoint union  $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$  of symbols  $c \in \mathcal{C}$ , called *constructors* and symbols  $f \in \mathcal{D}$ , called *defined functions*, where

$\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} - \mathcal{D}$ .

### Context-sensitive rewriting.

A mapping  $\mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N})$  is a *replacement map* (or  $\mathcal{F}$ -map) if  $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$  [10]. Let  $M_{\mathcal{F}}$  be the set of all  $\mathcal{F}$ -maps (or  $M_{\mathcal{R}}$  for the  $\mathcal{F}$ -maps of a TRS  $(\mathcal{F}, R)$ ). A binary relation  $R$  on terms is  $\mu$ -monotonic if  $t R s$  implies  $f(t_1, \dots, t_{i-1}, t, \dots, t_k) R f(t_1, \dots, t_{i-1}, s, \dots, t_k)$  for every  $t, s, t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . The set of  $\mu$ -replacing positions  $\text{Pos}^{\mu}(t)$  of  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is:  $\text{Pos}^{\mu}(t) = \{\Lambda\}$ , if  $t \in \mathcal{X}$  and  $\text{Pos}^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(\text{root}(t))} i. \text{Pos}^{\mu}(t_i)$ , if  $t \notin \mathcal{X}$ . The set of replacing variables of  $t$  is  $\text{Var}^{\mu}(t) = \{x \in \text{Var}(t) \mid \exists p \in \text{Pos}^{\mu}(t), t|_p = x\}$ . The  $\mu$ -replacing subterm relation  $\triangleright_{\mu}$  is given by  $t \triangleright_{\mu} s$  if there is  $p \in \text{Pos}^{\mu}(t)$  such that  $s = t|_p$ . We write  $t \triangleright_{\mu} s$  if  $t \triangleright_{\mu} s$  and  $t \neq s$ . In *context-sensitive rewriting* (CSR [10]), we (only) contract replacing redexes:  $t$   $\mu$ -rewrites to  $s$ , written  $t \hookrightarrow_{\mu} s$  (or  $t \hookrightarrow_{\mathcal{R}, \mu} s$ ), if  $t \xrightarrow{p}_{\mathcal{R}} s$  and  $p \in \text{Pos}^{\mu}(t)$ . A TRS  $\mathcal{R}$  is  $\mu$ -terminating if  $\hookrightarrow_{\mu}$  is terminating. A term  $t$  is  $\mu$ -terminating if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \hookrightarrow_{\mu} t_2 \hookrightarrow_{\mu} \dots \hookrightarrow_{\mu} t_n \hookrightarrow_{\mu} \dots$  starting from  $t$ . A pair  $(\mathcal{R}, \mu)$  where  $\mathcal{R}$  is a TRS and  $\mu \in M_{\mathcal{R}}$  is often called a CS-TRS.

### Dependency pairs.

Given a TRS  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  a new TRS  $\text{DP}(\mathcal{R}) = (\mathcal{F}^{\sharp}, D(R))$  of *dependency pairs* for  $\mathcal{R}$  is given as follows: if  $f(t_1, \dots, t_m) \rightarrow r \in R$  and  $r = C[g(s_1, \dots, s_n)]$  for some defined symbol  $g \in \mathcal{D}$  and  $s_1, \dots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , then  $f^{\sharp}(t_1, \dots, t_m) \rightarrow g^{\sharp}(s_1, \dots, s_n) \in D(R)$ , where  $f^{\sharp}$  and  $g^{\sharp}$  are new fresh symbols (called *tuple symbols*) associated to defined symbols  $f$  and  $g$  respectively [1]. Let  $\mathcal{D}^{\sharp}$  be the set of tuple symbols associated to symbols in  $\mathcal{D}$  and  $\mathcal{F}^{\sharp} = \mathcal{F} \cup \mathcal{D}^{\sharp}$ . As usual, for  $t = f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^{\sharp}$  to denote the *marked term*  $f^{\sharp}(t_1, \dots, t_k)$ . Conversely, given a marked term  $t = f^{\sharp}(t_1, \dots, t_k)$ , where  $t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^{\flat}$  to denote the term  $f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . Given  $T \subseteq \mathcal{T}(\mathcal{F}, \mathcal{X})$ , let  $T^{\sharp}$  be the set  $\{t^{\sharp} \mid t \in T\}$ .

## 3 Structure of infinite $\mu$ -rewrite sequences

Let  $\mathcal{M}_{\infty, \mu}$  be a set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict  $\mu$ -replacing subterm  $s$  of  $t$  (i.e.,  $t \triangleright_{\mu} s$ ) is  $\mu$ -terminating. Obviously, if  $t \in \mathcal{M}_{\infty, \mu}$ , then  $\text{root}(t)$  is a defined symbol. Furthermore, since  $\mu$ -terminating terms are preserved under  $\mu$ -rewriting, it follows that  $\mathcal{M}_{\infty, \mu}$  is also preserved under *inner*  $\mu$ -rewritings.

**Lemma 3.1** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $t \in \mathcal{M}_{\infty, \mu}$ . If  $t \xrightarrow{> \epsilon}^* s$ , then  $s \in \mathcal{M}_{\infty, \mu}$ .*

The following proposition establishes that, given  $t \in \mathcal{M}_{\infty, \mu}$ , there are two ways for an infinite  $\mu$ -rewrite sequence to proceed. The first one is by using ‘visible’ parts of the rules which correspond to  $\mu$ -replacing subterms in the right-hand sides which

are rooted by a defined symbol. The second one is by showing up ‘hidden’ non- $\mu$ -terminating subterms which are activated by *migrating* variables in a rule  $l \rightarrow r$ , i.e., variables  $x \in \text{Var}^\mu(r) - \text{Var}^\mu(l)$  which are *not*  $\mu$ -replacing in the left-hand side  $l$  but become  $\mu$ -replacing in the right-hand side  $r$ .

**Proposition 3.2** [3] *Let  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . For all  $t \in \mathcal{M}_{\infty, \mu}$ , there exist  $l \rightarrow r \in R$ , a substitution  $\sigma$  and a term  $u \in \mathcal{M}_{\infty, \mu}$  such that  $t \xrightarrow{>^*} \sigma(l) \xrightarrow{<} \sigma(r) \triangleright_\mu u$  and either (1) there is a  $\mu$ -replacing subterm  $s$  of  $r$  such that  $u = \sigma(s)$ , or (2) there is  $x \in \text{Var}^\mu(r) - \text{Var}^\mu(l)$  such that  $\sigma(x) \triangleright_\mu u$ .*

Now we investigate the structure of such sequences in more detail. In the following, we write  $t \triangleright_\mu s$  to denote that  $s$  is a non-replacing (hence strict!) subterm of  $t$ :  $t \triangleright_\mu s$  if there is  $p \in \text{Pos}(t) - \text{Pos}^\mu(t)$  such that  $s = t|_p$ .

**Definition 3.3** [Hidden symbol] Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . We say that  $f \in \mathcal{F}$  is a *hidden symbol* if there is a rule  $l \rightarrow r \in R$  and  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $r \triangleright_\mu t$  and  $\text{root}(t) = f$ . Let  $\mathcal{H}(\mathcal{R}, \mu)$  (or just  $\mathcal{H}$ , if  $\mathcal{R}$  and  $\mu$  are clear for the context) be the set of all hidden symbols in  $(\mathcal{R}, \mu)$ .

**Lemma 3.4** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $\sigma$  be a substitution. If there is a rule  $l \rightarrow r \in R$  such that  $\sigma(l) \not\triangleright t$  and  $\sigma(r) \triangleright_\mu t$ , then there is no  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$ . Furthermore, there is a term  $t' \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $r \triangleright_\mu t'$ ,  $\sigma(t') = t$  and  $\text{root}(t) = \text{root}(t') \in \mathcal{H}$ .*

**Proof.** By contradiction. If there is  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$ , then since variables in  $l$  are always below some function symbol we have  $\sigma(l) \triangleright t$ , leading to a contradiction.

Since there is no  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$  but we have that  $\sigma(r) \triangleright_\mu t$ , then there is a non-variable and non-replacing position  $p \in \text{Pos}_{\mathcal{F}}(r) - \text{Pos}^\mu(r)$ , such that  $\text{root}(r|_p) = \text{root}(t) \in \mathcal{H}(\mathcal{R}, \mu)$  and  $\sigma(r|_p) = t$ . Then, we let  $t' = r|_p$ .  $\square$

The following lemma establishes that minimal non- $\mu$ -terminating and non- $\mu$ -replacing subterms occurring in a  $\mu$ -rewrite sequence involving only minimal terms directly come from the first term in the sequence or are rooted by a hidden symbol.

**Lemma 3.5** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\mathcal{A}$  be a finite  $\mu$ -rewrite sequence  $t_1 \hookrightarrow t_2 \hookrightarrow \dots \hookrightarrow t_n$  with  $t_i \in \mathcal{M}_{\infty, \mu}$  for all  $i$ ,  $1 \leq i \leq n$  and  $n \geq 1$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_1 \not\triangleright t$  and  $t_n \triangleright_\mu t$ , then  $\text{root}(t) \in \mathcal{H}$ .*

**Proof.** By induction on  $n$ :

- (i) If  $n = 1$ , then it is vacuously true.
- (ii) If  $n > 1$ , then we assume that  $t_1 \not\triangleright t$  and  $t_n \triangleright_\mu t$ . Let  $l \rightarrow r \in R$  be such that  $t_{n-1} = C[\sigma(l)]$  and  $t_n = C[\sigma(r)]$  for some context  $C[\ ]$ . We consider two cases: either  $t_{n-1} \triangleright_\mu t$  holds or not.
  - (a) If  $t_{n-1} \triangleright_\mu t$ , then by the induction hypothesis we have that  $\text{root}(t) \in \mathcal{H}$ .
  - (b) If  $t_{n-1} \triangleright_\mu t$  does not hold, then one of the following cases holds:
    - (1)  $t_{n-1} \triangleright_\mu t$ ; then  $t_{n-1} \in \mathcal{M}_{\infty, \mu}$  implies that  $t \notin \mathcal{M}_{\infty, \mu}$ , leading to a contradiction.

- (2)  $t_{n-1} \not\triangleright_{\mu} t$  (in particular,  $\sigma(l) \not\triangleright_{\mu} t$ ); then, since  $t_n \triangleright_{\mu} t$  there must be  $\sigma(r) \triangleright_{\mu} t$ . Thus, by Lemma 3.4 we conclude that  $\text{root}(t) \in \mathcal{H}$ .  $\square$

Now we use the previous lemmas to investigate infinite sequences that mix  $\mu$ -rewriting steps on minimal non- $\mu$ -terminating terms and the extraction of such subterms as  $\mu$ -replacing subterms of (instances of) right-hand sides of rules.

**Proposition 3.6** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\mathcal{A}$  be an infinite sequence of the form  $t_1 \xrightarrow{\epsilon} s_1 \triangleright_{\mu} t'_2 \xrightarrow{\epsilon^*} t_2 \xrightarrow{\epsilon} s_2 \triangleright_{\mu} t'_3 \xrightarrow{\epsilon^*} t_3 \cdots$  with  $t_i, t'_i \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_i \triangleright_{\mu} t$  for some  $i \geq 1$ , then  $\text{root}(t) \in \mathcal{H} \cap \mathcal{D}$  or  $t_1 \triangleright_{\mu} t$ .*

**Proof.** By induction on  $i$ :

- (i) If  $i = 1$ , it is trivial.
- (ii) If  $i > 1$  and  $t_i \triangleright_{\mu} t$ , then we consider two cases: either  $t_{i-1} \triangleright_{\mu} t$  holds or not.
  - (a) If  $t_{i-1} \triangleright_{\mu} t$ , then by the induction hypothesis we get  $t_1 \triangleright_{\mu} t$  or  $\text{root}(t) \in \mathcal{H} \cap \mathcal{D}$ , as desired.
  - (b) If  $t_{i-1} \not\triangleright_{\mu} t$  does not hold, then let  $l \rightarrow r \in R$  be such that  $t_{i-1} = \sigma(l)$  and  $s_{i-1} = \sigma(r) \triangleright_{\mu} t'_i$ . We consider two cases:
    - (1) if  $t_{i-1} \triangleright_{\mu} t$  then being  $t_{i-1} \in \mathcal{M}_{\infty, \mu}$  it would imply that  $t \notin \mathcal{M}_{\infty, \mu}$ , thus leading to a contradiction.
    - (2) If  $t_{i-1} \not\triangleright_{\mu} t$ , then we consider two cases: either  $t'_i \triangleright_{\mu} t$  or  $t'_i \not\triangleright_{\mu} t$ .
      - (A) If  $t'_i \triangleright_{\mu} t$ , since  $t'_i, t \in \mathcal{M}_{\infty, \mu}$  the case  $t'_i \triangleright_{\mu} t$  is excluded and the only possibility is that  $t'_i \triangleright_{\mu} t$ . Then, since  $\sigma(l) = t_{i-1} \not\triangleright_{\mu} t$  and  $\sigma(r) \triangleright_{\mu} t'_i \triangleright_{\mu} t$ , i.e.  $\sigma(r) \triangleright_{\mu} t$ , by Lemma 3.4 we conclude that  $\text{root}(t) \in \mathcal{H}$ . Since  $t \in \mathcal{M}_{\infty, \mu}$ , we have  $\text{root}(t) \in \mathcal{H} \cap \mathcal{D}$ .
      - (B) If  $t'_i \not\triangleright_{\mu} t$ , then, by applying Lemma 3.1 and Lemma 3.5 to the  $\mu$ -rewrite sequence  $t'_i \xrightarrow{\epsilon^*} t_i$  we conclude  $\text{root}(t) \in \mathcal{H} \cap \mathcal{D}$ .  $\square$

As an immediate consequence of Proposition 3.6, we have the following result which we will use later.

**Corollary 3.7** *Let  $(\mathcal{R}, \mu)$  be a CS-TRS,  $\mathcal{A}$  be an infinite sequence of the form  $t_1 \xrightarrow{\epsilon} s_1 \triangleright_{\mu} t'_2 \xrightarrow{\epsilon^*} t_2 \xrightarrow{\epsilon} s_2 \triangleright_{\mu} t'_3 \xrightarrow{\epsilon^*} t_3 \cdots$  with  $t_i, t'_i \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_i \triangleright_{\mu} t$  for some  $i \geq 1$  and  $\text{root}(t) \in \mathcal{D} - \mathcal{H}$ , then  $t_1 \triangleright_{\mu} t$ .*

## 4 Revised context-sensitive dependency graph

Proposition 3.2 motivates the definition of context-sensitive dependency pair(s) and chain of context-sensitive dependency pairs.

**Definition 4.1** [CS-dependency pairs [3]] Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . We define  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to be the set of

context-sensitive dependency pairs (CS-DPs) where:

$$\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) = \{l^{\sharp} \rightarrow s^{\sharp} \mid l \rightarrow r \in R, r \succeq_{\mu} s, \text{root}(s) \in \mathcal{D}, l \not\prec_{\mu} s\}$$

and  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \{l^{\sharp} \rightarrow x \mid l \rightarrow r \in R, x \in \text{Var}^{\mu}(r) - \text{Var}^{\mu}(l)\}$ . We extend  $\mu \in M_{\mathcal{F}}$  into  $\mu^{\sharp} \in M_{\mathcal{F}^{\sharp}}$  by  $\mu^{\sharp}(f) = \mu(f)$  if  $f \in \mathcal{F}$ , and  $\mu^{\sharp}(f^{\sharp}) = \mu(f)$  if  $f \in \mathcal{D}$ .

**Example 4.2** Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 1.2. There are six context-sensitive dependency pairs:

```

1: PRIMES -> SIEVE(from(s(s(0))))
2: PRIMES -> FROM(s(s(0)))
3: TAIL(cons(X,Y)) -> Y
4: IF(true,X,Y) -> X
5: IF(false,X,Y) -> Y
6: FILTER(s(s(X)),cons(Y,Z)) ->
   IF(divides(s(s(X)),Y),filter(s(s(X)),Z),cons(Y,filter(X,sieve(Y))))

```

Note the three collapsing dependency pairs: (3), (4), and (5).

**Definition 4.3** [Chain of CS-DPs [3]] Let  $(\mathcal{R}, \mu)$  be a CS-TRS. Given  $\mathcal{P} \subseteq \text{DP}(\mathcal{R}, \mu)$ , an  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , for  $i \geq 1$  such that there is a substitution  $\sigma$  satisfying both:

- (i)  $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
- (ii) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is  $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) \succeq_{\mu} s_i$  and  $s_i^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u_{i+1})$ .

Here, as usual we assume that different occurrences of dependency pairs do not share any variable (renamings are used if necessary). An  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chain is called *minimal* if for all  $i \geq 1$   $\sigma(u_i)^{\sharp} \in \mathcal{M}_{\infty, \mu}$ ,  $s_i \in \mathcal{M}_{\infty, \mu}$  (whenever they occur in the chain) and all dependency pairs in  $\mathcal{P}$  occur infinitely often.

**Remark 4.4** When an  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^{\sharp})$ -chain is written for a given substitution  $\sigma$ , we write  $\sigma(u) \xrightarrow{\text{DP}(\mathcal{R}, \mu), \mu^{\sharp}} \sigma(v)$  for steps which use a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  but we rather write  $\sigma(u) \xrightarrow{\text{DP}(\mathcal{R}, \mu), \mu^{\sharp}} s^{\sharp}$  for steps which use a dependency pair  $u \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , where  $s$  is as in Definition 4.3.

**Theorem 4.5 (Correctness and completeness [3])** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ .  $\mathcal{R}$  is  $\mu$ -terminating if and only if there is no infinite  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^{\sharp})$ -chain.

An essential aspect of the mechanization of the dependency pairs approach is the analysis of infinite sequences of dependency pairs by looking at (the cycles  $\mathfrak{C}$  of) the *dependency graph* associated to the TRS  $\mathcal{R}$ . In [3], the *context-sensitive dependency graph*, is defined as follows:

- (i) There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  to a dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if there is a substitutions  $\sigma$  such that  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu^{\sharp}} \sigma(u')$ .
- (ii) There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to each dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$ .

Connecting each collapsing dependency pair with *every* other dependency pair

makes the cycles bigger, thus making some proofs of termination harder. Thanks to the results in the previous section, we can prove the following.

**Theorem 4.6** *There is no infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain involving an infinite number of dependency pairs  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  such that  $\text{root}(u_{i+1})^\sharp \notin \mathcal{H}$ .*

**Proof.** By contradiction. Let  $A$  be an infinite  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -minimal chain of CS-DPs characterized by the CS-DPs  $u_i \rightarrow v_i$  for  $i \geq 1$ :

$$\sigma(u_1) \xrightarrow{\epsilon} \mathcal{P} s_1^\sharp \xrightarrow{\epsilon} \mathcal{R}^* \sigma(u_2) \xrightarrow{\epsilon} \mathcal{P} s_2^\sharp \xrightarrow{\epsilon} \mathcal{R}^* \dots$$

where,  $s_i^\sharp = \sigma(v_i)$  if  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  and  $\sigma(x_i) \triangleright_\mu s_i$  if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . Let  $I$  be the infinite set of indices satisfying that for all  $i \in I$ ,  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  and  $\text{root}(u_{i+1})^\sharp \notin \mathcal{H}$ . Given  $i \in I$ , let  $\eta(i)$  be the ‘next’ positive integer in  $I$ :  $\eta(i) = \min(\{j \in I \mid j > i\})$ . Obviously, for all  $i \in I$ ,  $\eta(i) \in I$ . Now consider the following sequence  $A^\sharp$  which is obtained from  $A$  by ‘unsharpening’ the tuple symbols and using the rules  $l_i \rightarrow r_i$  which originate the dependency pairs  $u_i \rightarrow v_i$  which are used in  $A$ :

$$\sigma(u_1)^\sharp \xrightarrow{\epsilon} \mathcal{R} \sigma(r_1) \triangleright_\mu s_1 \xrightarrow{\epsilon} \mathcal{R}^* \sigma(u_2)^\sharp \xrightarrow{\epsilon} \mathcal{R} \sigma(r_2) \triangleright_\mu s_2 \xrightarrow{\epsilon} \mathcal{R}^* \dots$$

which corresponds to  $A$  above: by minimality of  $A$  (see Definition 4.3), we have that  $\sigma(u_i)^\sharp, s_i \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . By definition of  $I$ , for all  $i \in I$ ,  $v_i = x_i \in \mathcal{X}$ . By definition of collapsing dependency pair,  $x_i \in \text{Pos}(u_i) - \text{Pos}^\mu(u_i)$  and  $\sigma(x_i) \triangleright_\mu s_i$ . Thus,  $\sigma(u_i) \triangleright_{\mu^\sharp} s_i$  for all  $i \in I$ . By repeatedly applying Corollary 3.7, we have that  $\sigma(u_i) \triangleright_{\mu^\sharp} \sigma(u_{\eta(i)})$ , i.e.,  $\sigma(u_i) \triangleright \sigma(u_{\eta(i)})$  for all  $i \in I$ . Thus, we obtain an infinite  $\triangleright$ -sequence which contradicts well-foundedness of  $\triangleright$ .  $\square$

As a consequence of this result, we can dismiss the arcs of the dependency graph which connect collapsing dependency pairs  $u \rightarrow v$  and dependency pairs  $u' \rightarrow v'$  such that  $\text{root}(u')^\sharp \notin \mathcal{H}$ . This leads to a new definition of the context-sensitive dependency graph:

**Definition 4.7** [Context-Sensitive Dependency Graph] Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The context-sensitive dependency graph consists of the set  $\text{DP}(\mathcal{R}, \mu)$  of context-sensitive dependency pairs together with arcs which connect them as follows:

- (i) There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  to a dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if there is a substitutions  $\sigma$  such that  $\sigma(v) \xrightarrow{\epsilon} \mathcal{R}, \mu^\sharp \sigma(u')$ .
- (ii) There is an arc from a dependency pair  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to a dependency pair  $u' \rightarrow v' \in \text{DP}(\mathcal{R}, \mu)$  if  $\text{root}(u')^\sharp \in \mathcal{H}(\mathcal{R}, \mu)$ .

**Example 4.8** Consider again the TRS  $\mathcal{R}$  in Example 1.2. The hidden defined symbols are **filter**, **from** and **sieve**. The dependency graph which corresponds to this example is shown in Figure 1 (right). Note that, in contrast to the situation

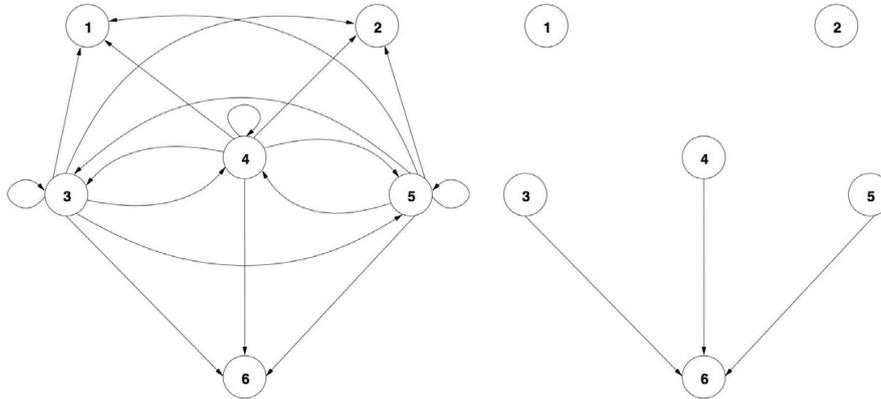


Fig. 1. Dependency graphs for Example 1.2: according to [3] (left) and according to Definition 4.7 (right)

with the old dependency graph (Figure 1, left) the new dependency graph has *no* cycle!

As noticed by Arts and Giesl, the presence of an infinite chain of dependency pairs corresponds to a cycle in the dependency graph (but not vice-versa). In the dependency graph this is true in the following sense: for each infinite chain of dependency pairs there is a suffix of the chain which corresponds to a cycle in the new dependency graph.

On the other hand, the treatment of cycles of the context-sensitive dependency graph for concluding termination by means of orderings remains as described in [3], but using the dependency graph in Definition 4.7.

**Example 4.9** Consider the following TRS  $\mathcal{R}$  [16, Example 4]

```
f(X) -> cons(X, f(g(X)))
g(0) -> s(0)
g(s(X)) -> s(s(g(X)))
sel(0, cons(X, Y)) -> X
sel(s(X), cons(Y, Z)) -> sel(X, Z)
```

with  $\mu(0) = \emptyset$ ,  $\mu(f) = \mu(g) = \mu(s) = \mu(\text{cons}) = \{1\}$ , and  $\mu(\text{sel}) = \{1, 2\}$ . Then,  $\text{DP}(\mathcal{R}, \mu)$  is:

```
G(s(X)) -> G(X)
SEL(s(X), cons(Y, Z)) -> SEL(X, Z)
SEL(s(X), cons(Y, Z)) -> Z
```

The set of hidden symbols is  $\mathcal{H} = \{f, g\}$  and there are two cycles:

- (i)  $G(s(X)) \rightarrow G(X)$
- (ii)  $\text{SEL}(s(X), \text{cons}(Y, Z)) \rightarrow \text{SEL}(X, Z)$

By using the subterm criterion [3, Section 5] we can easily prove that the system is  $\mu$ -terminating.

## 5 Narrowing context-sensitive dependency pairs

There are examples where the automation of the CS-DP method can fail or be more difficult due to the *estimation* of the arcs that connect two CS-dependency pairs (by means of functions  $\text{CAP}^\mu$  and  $\text{REN}^\mu$ , see [3]).

**Example 5.1** Consider the following example [13, Proposition 7]

$$\begin{aligned} f(0) &\rightarrow \text{cons}(0, f(s(0))) \\ f(s(0)) &\rightarrow f(p(s(0))) \\ p(s(x)) &\rightarrow x \end{aligned}$$

together with  $\mu(f) = \mu(p) = \mu(s) = \mu(\text{cons}) = \{1\}$  and  $\mu(0) = \emptyset$ . Then  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\begin{aligned} F(s(0)) &\rightarrow F(p(s(0))) \\ F(s(0)) &\rightarrow P(s(0)) \end{aligned}$$

The *estimated* CS-dependency graph contains one cycle consisting of the CS-dependency pair

$$F(s(0)) \rightarrow F(p(s(0)))$$

However, this cycle does *not* belong to the CS-dependency graph because there is no way to  $\mu$ -rewrite  $F(p(s(0)))$  into  $F(s(0))$ !

The problem is that with the estimated CS-dependency graph, we connect more dependency pairs than needed. The over-estimation eventually comes when a CS-dependency pair  $u \rightarrow v$  is connected to  $u' \rightarrow v'$  in the estimated dependency graph and  $v$  and  $u'$  do not unify, i.e. at least a rewriting step with some rule of  $\mathcal{R}$  is needed to reduce (some instance of)  $v$  to (the corresponding instance of)  $u'$ . It is then possible that, after performing such a necessary  $\mu$ -rewriting step, the connection between them gets clearly lost, i.e. the nodes were not really connected in the graph. This is missed in the estimated dependency graph due to the use of  $\text{CAP}^\mu$  and  $\text{REN}^\mu$ . We can use *context-sensitive* narrowing to avoid this problem.

**Definition 5.2** [Context-sensitive narrowing [10]] Let  $(\mathcal{R}, \mu)$  be a CS-TRS. A term  $t$   $\mu$ -narrows to a term  $s$  (written  $t \rightsquigarrow_\mu s$ ), if there exists a non-variable position  $p \in \text{Pos}^\mu(t)$ ,  $\theta$  is the most general unifier of  $t|_p$  and  $l$  for a rewrite rule  $l \rightarrow r$  in  $\mathcal{R}$  (sharing no variable with  $t$ ), and  $s = \theta(t[r]_p)$ .

To achieve more precision when connecting two CS-DPs in a  $(\mathcal{R}, \text{DP}(\mathcal{R}, \mu), \mu^\sharp)$ -chain, we may perform all possible  $\mu$ -narrowings steps on  $v$  in order to develop the reductions from (instances of)  $v$  to (instances of)  $u'$ . Then, we obtain new terms  $v_1, \dots, v_n$  which are  $\mu$ -narrowings of  $v$  with unifier  $\theta_i$  for  $i \in \{1, \dots, n\}$  and can be used instead of  $v$ . Not only the right-hand sides of the CS-dependency pairs are  $\mu$ -narrowed: the unifier which used in the narrowing step should also be applied on the left-hand sides of the  $\mu$ -narrowed pairs. Therefore, we can replace a CS-dependency pair  $u \rightarrow v$  by all new  $\mu$ -narrowed pairs  $\theta_1(u) \rightarrow v_1, \dots, \theta_n(u) \rightarrow v_n$ . The next result shows that under those conditions, the set of CS-dependency pairs can be replaced by their narrowings without losing correctness or completeness.

**Theorem 5.3 (Narrowing refinement for CS-termination)** *Let  $\mathcal{R}$  be a TRS and let  $\mathcal{P}$  be a set of CS-dependency pairs. Let  $u \rightarrow v \in \mathcal{P}$  such that  $v$  is linear and for all  $u' \rightarrow v' \in \mathcal{P}$  (with renamed variables) the terms  $v$  and  $u'$  are not unifiable. Let*

$$\mathcal{P}' = (\mathcal{P} - \{u \rightarrow v\}) \cup \{u' \rightarrow v' \mid u' \rightarrow v' \text{ is a narrowing of } u \rightarrow v\}.$$

*There exists an infinite  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain iff there exists an infinite  $(\mathcal{R}, \mathcal{P}', \mu^\sharp)$ -chain.*

**Proof.** The proof of this theorem corresponds to the proof of Theorem 25 in [1]. Note that only dependency pairs in  $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  can be narrowed. As in Arts and Giesl's proof, requiring the no-unification between the CS-dependency pair to narrow and the rest of the set; the linearity of  $v$ ; and the renaming of the variables of the different (occurrences of) dependency pairs is still necessary to guarantee that narrowing CS-dependency pairs do not miss any chain from  $\mathcal{P}$ . The main difference is that the reductions between dependency pairs are  $\mu$ -reductions, but since we are using  $\mu$ -narrowing, the whole proof is adapted without loss of generality.  $\square$

Thus, after narrowing the dependency pairs in  $\text{DP}(\mathcal{R}, \mu)$  we can build a *narrowed* dependency graph. Afterwards, we can use it to check termination as usual.

**Example 5.4** (Continuing Example 5.1) Since the right-hand side of the CS-dependency pair in Example 5.1 does not unify with any left-hand side of a dependency pair, (including itself) and it can be  $\mu$ -narrowed at position 1 (notice that  $\mu(\mathbf{f}) = \{1\}$ ) by using the rule

$$p(s(X)) \rightarrow X$$

we can replace it by its  $\mu$ -narrowed CS-dependency pair:

$$F(s(0)) \rightarrow F(0)$$

The *narrowed* pair does not form any cycle in the estimated narrowed graph and termination is easily proved now.

## 6 Experiments

The techniques described in the previous sections have been implemented as part of the tool MU-TERM [2,12]. We have used our new implementation to compare with the last version of the tool: MU-TERM 4.3. The benchmarks were executed in a completely automatic way (see [2] for a description of MU-TERM's termination expert) and with a timeout of 1 minute on the 90 examples in the Context-Sensitive Rewriting subcategory of the 2006 Termination Competition, available through the URL:

<http://www.lri.fr/~marche/termination-competition/2006>

As remarked above, our termination expert works as explained in [2] for version 4.3 of MU-TERM. For the new version 4.4 of MU-TERM, we have just used the new definition of the (eventually narrowed) dependency graph. We have compared our new implementation with the previous version of MU-TERM (corresponding to [3]).

Termination Tool	Total	CS-DPs	CSRPO	Transf.	Average time
MU-TERM (PROLE'06)	66	65	0	1	1.68s
MU-TERM (FST&TCS'06)	56	45	7	4	4.55s
AProVE	56	0	0	56	4.74s

Table 1

We have also used AProVE for proving termination of the examples. AProVE [9] is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques. AProVE is also able to prove termination of Context-Sensitive Rewriting by using *transformations*. Such transformations obtain a proof of the  $\mu$ -termination of a TRS  $\mathcal{R}$  as a proof of termination of a transformed TRS  $\mathcal{R}_\Theta^\mu$  (where  $\Theta$  represents the transformation). If we are able to prove *termination* of  $\mathcal{R}_\Theta^\mu$  (using the standard methods), then the  $\mu$ -termination of  $\mathcal{R}$  is ensured (see [13] for a recent survey).

A complete account of our experiments can be found here:

<http://www.dsic.upv.es/~rgutierrez/muterm/prole/benchmarks.html>

Table 1 summarizes our benchmarks. As shown in Table 1, the results make clear the advantages of the new refinement: we are able to prove 10 additional examples and the proofs are almost *three* times faster (in the average).

Furthermore, we can say that the new refinement developed for the CS-DP approach greatly improves on the use of other techniques: the use of transformations and other (also powerful) techniques like CSRPO [4] becomes now anecdotic or null.

## 7 Conclusions

We have introduced a simplification of the context-sensitive dependency graph by restricting the outgoing links of collapsing dependency pairs to dependency pairs headed by the so-called *hidden* symbols. Hidden symbols are defined symbols that occur in non-replacing positions in the right-hand sides of some rule in the TRS. This greatly improves the performance of termination proofs based on the dependency graph proposed in [3]. Narrowing context-sensitive dependency pairs has also been investigated. It can also be helpful to simplify or restructure the dependency graph and eventually simplify the proof of termination. Regarding the practical use of the (refinements on the) new CS-dependency graph in proofs of termination of *CSR*, we have implemented these ideas as part of the termination tool MU-TERM and we have obtained quite good results in terms of new examples which could be proved, and also regarding the time for achieving the proofs.

Since the state-of-the-art of DP-based techniques for proving termination of *CSR* which has been introduced in this paper corresponds to the development of DPs in the late nineties, we can conclude that further improvements of CS-DPs will evolve in such a way that the CS-dependency pairs approach can play for *CSR* the (practical and theoretical) role than dependency pairs play in rewriting.

Many other aspects of the dependency pairs approach are also worth to be considered and extended to *CSR* (modularity issues, innermost computations, usable rules, . . .). They provide an interesting subject for future work.

## References

- [1] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
- [2] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science*, to appear, 2007.
- [3] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In N. Garg and S. Arun-Kumar, editors *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.
- [4] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.
- [5] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Sestoft and N. Heintze, editors, *Proc. of PEPM'04*, pages 147-158, ACM Press, New York, 2004.
- [6] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2006.
- [7] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271-285, Springer-Verlag, Berlin, 1999.
- [8] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379-427, 2004.
- [9] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281-286, Springer-Verlag, Berlin, 2006. Available at <http://www-i2.informatik.rwth-aachen.de/AProVE>.
- [10] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1-61, January 1998.
- [11] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293-343, 2002.
- [12] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of RTA'04*, LNCS 3091:200-209, Springer-Verlag, Berlin, 2004. Available at <http://www.dsic.upv.es/~slucas/csr/termination/muterm>.
- [13] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782-1846, 2006.
- [14] E. Ohlbusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.
- [15] TeReSe, editor, *Term Rewriting Systems*, Cambridge University Press, 2003.
- [16] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of RTA'97*, LNCS 1232:172-186, Springer-Verlag, Berlin, 1997.

## 8.8 Proving Termination of Context-Sensitive Rewriting with MU-TERM

3. B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. **Proving Termination of Context-Sensitive Rewriting with MU-TERM.** *Electronic Notes in Theoretical Computer Science*, 188:105–115, 2007.



ELSEVIER

Available online at [www.sciencedirect.com](http://www.sciencedirect.com)

Electronic Notes in Theoretical Computer Science 188 (2007) 105–115

Electronic Notes in  
Theoretical Computer  
Science[www.elsevier.com/locate/entcs](http://www.elsevier.com/locate/entcs)

# Proving Termination of Context-Sensitive Rewriting with MU-TERM

Beatriz Alarcón, Raúl Gutiérrez,  
José Iborra and Salvador Lucas<sup>1,2</sup>

*Departamento de Sistemas Informáticos y Computación  
Universidad Politécnica de Valencia  
Valencia, Spain*

---

## Abstract

Context-sensitive rewriting (CSR) is a restriction of rewriting which forbids reductions on selected arguments of functions. Proving termination of CSR is an interesting problem with several applications in the fields of term rewriting and programming languages. Several methods have been developed for proving termination of CSR. The new version of MU-TERM which we present here implements all currently known techniques. Furthermore, we show how to combine them to furnish MU-TERM with an *expert* which is able to automatically perform the termination proofs. Finally, we provide a first experimental evaluation of the tool.

*Keywords:* Context-sensitive rewriting, term rewriting, program analysis, termination.

---

## 1 Introduction

Restrictions of rewriting can eventually *achieve* termination of rewriting computations by pruning all infinite rewrite sequences issued from every term. However, such kind of improvements can be difficult to prove. Context-sensitive rewriting (*CSR* [17,18]) is a restriction of rewriting which is useful for describing semantic aspects of programming languages (e.g., Maude, OBJ2, OBJ3, or CafeOBJ) and analyzing termination of the corresponding programs (see [8,9,13,18,22] for further motivation). In *CSR*, a *replacement map*  $\mu$  discriminates, for each symbol of the signature, the argument positions  $\mu(f)$  on which rewritings are allowed. In this way, for a given Term Rewriting System (TRS), we obtain a restriction of the rewrite relation which

---

<sup>1</sup> This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and HA 2006-0007, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination ALA/95/23/2003/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.

<sup>2</sup> Email: {[balarcon,rgutierrez,jiborra,slucas](mailto:balarcon,rgutierrez,jiborra,slucas}@dsic.upv.es)}@dsic.upv.es

we call *context-sensitive rewriting*. A TRS  $\mathcal{R}$  together with a replacement map  $\mu$  is often called a CS-TRS and written  $(\mathcal{R}, \mu)$ .

Proving termination of *CSR* is an interesting problem with several applications in the fields of term rewriting and programming languages (see [22]). There are two main approaches to prove termination of a CS-TRS  $(\mathcal{R}, \mu)$ :

- *direct proofs* use adapted versions of term orderings such as RPOs and polynomial orderings to compare the left- and right-hand sides of the rules [4,11,20,21]; and
- *transformations* which obtain a transformed TRS  $\mathcal{R}_\Theta^\mu$  (where  $\Theta$  represents the transformation). If we are able to prove *termination* of  $\mathcal{R}_\Theta^\mu$  (using the standard methods), then termination of the CS-TRS is ensured (see [22] for a recent survey).

MU-TERM was the first tool implementing techniques for proving termination of *CSR* [19]. The tool is available here:

<http://www.dsic.upv.es/~slucas/csr/termination/muterm>

Nowadays, the tool AProVE [15] also accepts context-sensitive termination problems specified in the TPDB format<sup>3</sup>. However, AProVE's proofs of termination of *CSR* are based on using transformations (i.e., no direct proof method is currently available). The new version of MU-TERM which we present here implements all currently known techniques. The new contributions which we report in this paper are the following:

- (i) We have implemented the *context-sensitive recursive path ordering* described in [4].
- (ii) We have implemented the *context-sensitive dependency pairs approach* described in [2].
- (iii) On the basis of recent theoretical and experimental results (see [22]), we have developed a termination expert for *CSR* which combines the different existing techniques for proving termination of *CSR* without any interaction with the user.

Finally, we want to mention that the Maude Termination Tool [9]:

<http://www.lcc.uma.es/~duran/MTT>

which transforms proofs of termination of Maude programs into proofs of termination of *CSR* uses MU-TERM's expert as an auxiliary tool.

We assume a basic knowledge about term rewriting, see [24] for missing definitions and more information. In Section 2 we briefly describe the new features which have been added to MU-TERM. Section 3 discusses the termination expert. Section 4 provides an experimental evaluation of the new version of MU-TERM. Section 5 concludes and discusses future work.

<sup>3</sup> See <http://www.lri.fr/~marche/tpdb/format.html>

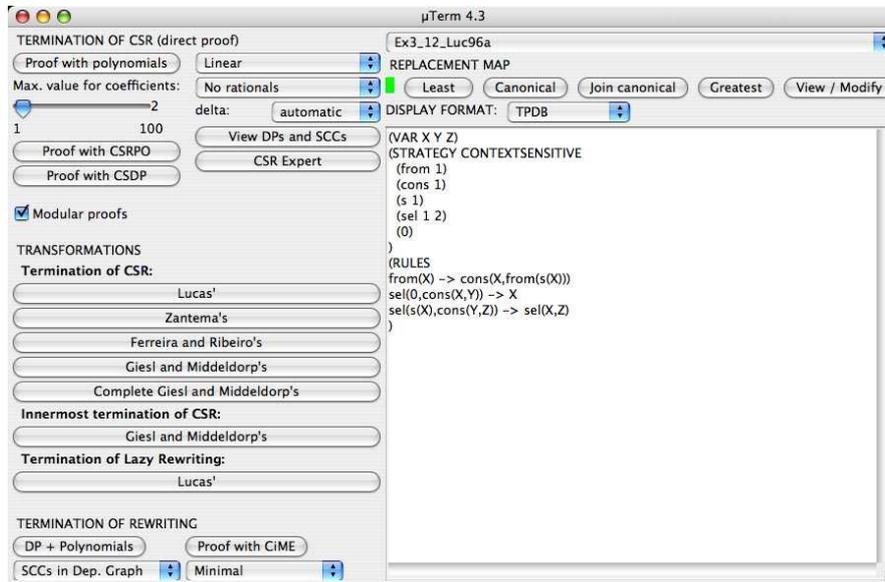


Fig. 1. Screenshot of the main window of MU-TERM 4.3

## 2 New features in MU-TERM

MU-TERM is written in Haskell<sup>4</sup>, and wxHaskell<sup>5</sup> has been used to develop the graphical user interface. The system consists of more than 45 Haskell modules containing more than 14000 lines of code. Compiled versions in several platforms (Linux, Mac OSX, and Windows) and instructions for the installation are available on the MU-TERM WWW site. A recent hybrid (Haskell/C#) version of the tool is also available for the .NET platform [3].

MU-TERM has a graphical user interface (see Figure 1) whose details (menu structure, supported formats, etc.) are given in [19]. Let us briefly recall the main features of the tool.

- **MODULARITY:** If the *modular proofs* are activated, then MU-TERM attempts a *safe* decomposition of the TRS in such a way that the components satisfy the modularity requirements described in [10]. If it succeeds in performing a non-trivial decomposition (i.e., MU-TERM obtains more than one component), then individual proofs of termination of *CSR* are attempted for each component.
- **DIRECT METHODS:** MU-TERM implements the use of polynomial interpretations as described in [20,21]. An interesting feature of MU-TERM is that it generates polynomial interpretations with *rational* coefficients.
- **TRANSFORMATIONS:** MU-TERM also implements a number of transformations for proving termination of *CSR* (see [13,22]).

<sup>4</sup> See <http://haskell.org/>.

<sup>5</sup> See <http://wxhaskell.sourceforge.net>.

In the following, we briefly describe the new features implemented in the current version of MU-TERM.

### 2.1 Context-Sensitive Recursive Path Ordering (CSRPO)

CSRPO extends the recursive path ordering (RPO [7]) to context-sensitive terms [4]. The first idea that comes in mind to extend RPO to *CSR* (CSRPO) is *marking* the symbols which are in blocked positions and consider them smaller than the active ones. Therefore, terms in blocked positions become smaller. However, marking all symbols in non-replacing positions can unnecessarily weaken the resulting ordering. Thus, in addition to the usual precedence<sup>6</sup>  $\succ_{\mathcal{F}}$  on the symbols of the signature  $\mathcal{F}$  of the TRS, a *marking map*, denoted by  $\mathbf{m}$ , is also used. The marking map defines, for every symbol and every blocked position, the set of symbols that should be marked. By  $\underline{\mathcal{F}}$  we denote the set of marked symbols corresponding to  $\mathcal{F}$ . Given a  $k$ -ary symbol  $f$  in  $\mathcal{F} \cup \underline{\mathcal{F}}$  and  $i \in \{1, \dots, k\}$ , a marking map  $\mathbf{m}$  provides the subset of symbols in  $\mathcal{F}$  that should be marked, i.e.  $\mathbf{m}(f, i) \subseteq \mathcal{F}$ . Marking maps are intended to mark *only* blocked arguments, i.e.,  $\mathbf{m}(f, i) = \emptyset$  if  $i \in \mu(f)$  for all  $f \in \mathcal{F}$ . In this way, we mark only the necessary symbols (in blocked positions), see [4] for a thorough discussion.

**Example 2.1** Consider the following TRS  $\mathcal{R}$ :

```

from(X) -> cons(X, from(s(X)))
sel(0, cons(X, Y)) -> X
sel(s(X), cons(Y, Z)) -> sel(X, Z)

```

together with  $\mu(\mathbf{cons}) = \mu(\mathbf{s}) = \mu(\mathbf{from}) = \{1\}$  and  $\mu(\mathbf{sel}) = \{1, 2\}$ . The  $\mu$ -termination of  $\mathcal{R}$  can be proved by the CSRPO induced by the following precedence and marking map (computed by MU-TERM, see Figure 2):

$$\mathbf{sel} \succ_{\mathcal{F}} \mathbf{from} \succ_{\mathcal{F}} \mathbf{cons} \succ_{\mathcal{F}} \mathbf{s}$$

$$\mathbf{m}(\mathbf{cons}, 2) = \mathbf{m}(\mathbf{cons}, 2) = \{\mathbf{from}\}, \quad \mathbf{m}(\mathbf{from}, 1) = \emptyset$$

and lexicographic status for all function symbols.

Although the  $\mu$ -termination of  $\mathcal{R}$  in Example 2.1 can be proved by using the following polynomial interpretation:

$$\begin{aligned} [\mathbf{from}](X) &= 3X + 2 & [\mathbf{cons}](X, Y) &= X + \frac{1}{3}Y & [0] &= 0 \\ [\mathbf{s}](X) &= 2X + 1 & [\mathbf{sel}](X, Y) &= 2X^2Y + X + Y + 1 \end{aligned}$$

the proof using CSRPO is much faster.

### 2.2 Context-Sensitive Dependency Pairs (CSDPs)

Recently, the *dependency pairs approach* [1], one of the most powerful techniques for proving termination of rewriting, has been generalized to be used in proofs of

<sup>6</sup> By a precedence, we mean a reflexive and transitive relation.

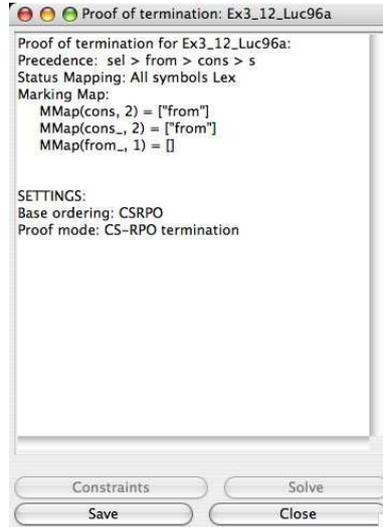


Fig. 2. Termination of CSR using CSRPO

termination of CSR [2].

Roughly speaking, given a TRS  $\mathcal{R}$ , the dependency pairs  $u \rightarrow v$  associated to  $\mathcal{R}$  conform a new TRS  $DP(\mathcal{R})$  which (together with  $\mathcal{R}$ ) determines the so-called *dependency chains* whose finiteness or infiniteness characterize termination of  $\mathcal{R}$ . The dependency pairs can be presented as a *dependency graph*, where the nodes of the graph are dependency pairs and the absence of infinite chains can be analyzed by considering the *cycles* in the graph. Two dependency pairs  $u \rightarrow v$  and  $u' \rightarrow v'$  in the graph are connected by an arc if there is a substitution  $\sigma$  which makes possible a (possibly empty) rewrite sequence (in  $\mathcal{R}$ ) from  $\sigma(v)$  to  $\sigma(u')$ . These ideas are generalized (with a number of non-trivial changes) to CSR.

**Example 2.2** Consider the following non-terminating TRS  $\mathcal{R}$  borrowing the well-known Toyama’s example [12, Example 1]:

$$f(a, b, X) \rightarrow f(X, X, X) \quad c \rightarrow a \quad c \rightarrow b$$

together with  $\mu(f) = \{3\}$ . The only dependency pair for this system is:

$$F(a, b, X) \rightarrow F(X, X, X)$$

where  $F$  is a ‘marked’ version (often called a *tuple symbol*) of  $f$  and we further assume that  $\mu(F) = \{3\}$ . It is not difficult to see that there is no substitution  $\sigma$  which is able to originate a (possibly empty) *context-sensitive* rewrite sequence (with  $\mathcal{R}$ !) from  $\sigma(F(X, X, X))$  to  $\sigma(F(a, b, X))$ . The replacement restriction  $\mu(F) = \{3\}$  is essential for this. Furthermore, this fact can be easily checked as explained in [2] and so it is implemented in MU-TERM.

A proof of  $\mu$ -termination of  $\mathcal{R}$  in Example 2.2 is *not* possible by using either CSRPO or polynomials with non-negative coefficients (see [11]). Also, as shown by Giesl and Middeldorp (see also [13]), among all the existing transformations for

Fig. 3. Termination of *CSR* using dependency pairs

proving termination of *CSR*, only the *complete* Giesl and Middeldorp's transformation [13] (yielding a TRS  $\mathcal{R}_C^\mu$ ) could be used in this case, but no concrete proof of termination for  $\mathcal{R}_C^\mu$  is known yet. Furthermore,  $\mathcal{R}_C^\mu$  has 13 dependency pairs and the dependency graph contains many cycles. In contrast, the CS-TRS has only *one* context-sensitive dependency pair and the corresponding dependency graph has *no* cycle! Thus, a direct and automatic proof of  $\mu$ -termination of  $\mathcal{R}$  is easy now (see Figure 3).

Although the subterms in the right-hand sides of the rules which are considered to build the context-sensitive dependency pairs are  $\mu$ -replacing terms, considering *only non-variable subterms* (as in Arts and Giesl's approach [1]) is *not* sufficient to obtain a correct approximation. As discussed in [2], in general we also need to consider dependency pairs with *variables* in the right-hand sides.

**Example 2.3** Consider the TRS  $\mathcal{R}$  [26, Example 5]:

$$\begin{array}{l} \text{if}(\text{true}, X, Y) \rightarrow X \quad \text{f}(X) \rightarrow \text{if}(X, c, \text{f}(\text{true})) \\ \text{if}(\text{false}, X, Y) \rightarrow Y \end{array}$$

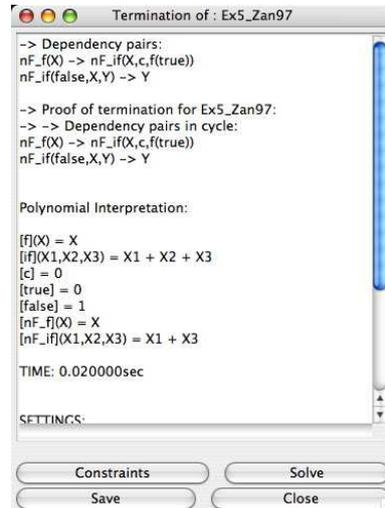
with  $\mu(\text{if}) = \{1, 2\}$ . There are two dependency pairs:

$$\begin{array}{l} \text{F}(X) \rightarrow \text{IF}(X, c, \text{f}(\text{true})) \\ \text{IF}(\text{false}, X, Y) \rightarrow Y \end{array}$$

with  $\mu$  extended by  $\mu(\text{F}) = \{1\}$  and  $\mu(\text{IF}) = \{1, 2\}$ .

A direct and automatic proof of  $\mu$ -termination of  $\mathcal{R}$  is possible with CSDPs by using an auxiliary polynomial ordering generated by a linear polynomial interpretation (computed by MU-TERM, see Figure 4).

A proof of  $\mu$ -termination of  $\mathcal{R}$  in Example 2.3 is *not* possible by using CSRPO. Furthermore, the  $\mu$ -termination of  $\mathcal{R}$  cannot be proved by using a polynomial or-

Fig. 4. Termination of *CSR* using dependency pairs and polynomials

dering based on a linear polynomial interpretation.

### 3 Automatically proving termination of *CSR* with MU-TERM

On the basis of recent theoretical and experimental results, we have developed a *termination expert* for *CSR* which combines the different existing techniques in a sequence of proof attempts which do not require any user interaction. The sequence of techniques which are tried by the expert is as follows:

- (i) Context-sensitive dependency pairs with auxiliary polynomial orderings based on polynomial interpretations using either:
  - (a) linear interpretations whose coefficients are taken from (1)  $\{0, 1\}$ , (2)  $\{0, 1, 2\}$ , or (3)  $\{0, \frac{1}{2}, 1, 2\}$ , in this order; or
  - (b) simple-mixed interpretations linear whose coefficients are taken from (1)  $\{0, 1\}$ , (2)  $\{0, 1, 2\}$ , or (3)  $\{0, \frac{1}{2}, 1, 2\}$ , again in this order.
- (ii) Context-sensitive recursive path ordering.
- (iii) Polynomial orderings generated from either linear or simple-mixed polynomial interpretations whose coefficients are rational numbers of the form  $\frac{p}{q}$  where  $0 \leq p, q \leq 5$  and  $q > 0$ .
- (iv) Transformations which obtain a TRS whose termination is proved by using the standard dependency pairs approach [1]. The transformations are attempted according to the decision tree in Figure 5 (explained below).

In the following, we motivate some of the choices we made for obtaining the concrete configuration of the previous sequence.

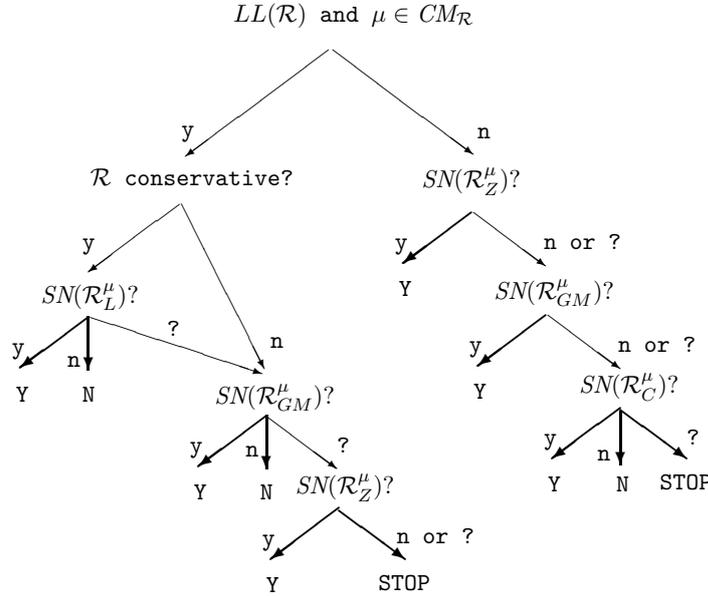


Fig. 5. Decision graph for proving termination of CSR by transformation

### 3.1 Use of polynomial interpretations

As shown in [21,23], the use of rational (or real) coefficients in polynomial interpretations can be helpful to achieve proofs of termination of (context-sensitive) rewriting. In this setting, in order to obtain a proof of  $\mu$ -termination of a TRS  $\mathcal{R} = (\mathcal{F}, R)$ , we use *parametric* polynomial interpretations for the symbols  $f \in \mathcal{F}$ , whose indeterminate coefficients are intended to be *real* (or *rational*) instead of natural or integer numbers. The termination problem is rephrased as a set of polynomial constraints on the indeterminate coefficients. This set of constraints is intended to be solved *in the domain of the real numbers*. Although such polynomial constraints over the reals are decidable [25], the difficulty of the procedure depends on the degree and composition of the parametric polynomials that we use for this. As in [6], we consider classes of polynomials which are well-suited for automatization of termination proofs: linear and simple-mixed polynomial interpretations.

The automatic generation of rational coefficients can be computationally expensive. For instance, MU-TERM manages rational (nonnegative) coefficients  $c \in \mathbb{Q}$  in polynomial interpretations as pairs numerator/denominator, i.e.,  $c = \frac{p}{q}$ , where  $p, q \in \mathbb{N}$  and  $q > 0$ . Thus, each rational coefficient  $c$  involves *two* integers. This leads to a huge search space in the corresponding constraint solving process [6,21]. For this reason, MU-TERM is furnished with three main *generation modes* [21]:

- (i) *No rationals*: here, no rational coefficient is allowed.
- (ii) *Rationals and integers*: here, since rational coefficients are intended to introduce non-monotonicity, we only use them with arguments  $i \notin \mu(f)$ .

- (iii) *All rationals*: where all coefficients of polynomials are intended to be rational numbers.

These generation modes are orderly used by the expert to try different polynomial interpretations.

Regarding the *range* of the coefficients, we follow the usual practice in similar termination tools, where coefficients are bounded to take values 0, 1, or 2 (see [6,15,16,27]). Note that (as in those related tools) this choice is *heuristic*, usually based on the experience. We do not know of any theoretical or empirical investigation which tries to guide the choice of appropriate bounds for the coefficients depending on the concrete termination problem. From our side, we just added the value  $\frac{1}{2}$  which enables a minimal (but still fruitful) use of rational coefficients. Again, these generation modes are orderly used by the expert to try different polynomial interpretations.

### 3.2 Use of transformations

In [22] we have investigated how to combine the different transformations for proving termination of *CSR*. Figure 5 provides a concrete decision tree for using the different transformations. Here,  $LL(\mathcal{R})$  means that  $\mathcal{R}$  is left-linear,  $CM_{\mathcal{R}}$  is the set of replacement maps which are not more restrictive than the *canonical* replacement map  $\mu_{\mathcal{R}}^{can}$  of the TRS  $\mathcal{R}$ . This replacement map has a number of interesting properties (see [17,18]) and can be automatically computed for each TRS (for instance, the tool MU-TERM can do that) thus giving the user the possibility of using *CSR* without explicitly introducing hand-crafted replacement restrictions. Finally,  $SN(\mathcal{R})?$  represents a check of *termination* of the TRS  $\mathcal{R}$ . More details can be found in [22].

## 4 Experimental evaluation

As remarked in the introduction, besides MU-TERM, AProVE is currently the only tool which is able to prove termination of *CSR* by using (non-trivial) transformations. AProVE is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques. AProVE implements a termination expert which successively tries different transformations for proving termination of *CSR* and uses a variety of different and complementary techniques for proving termination of rewriting, see [15,14]. We have considered the (Linux-based, completely automatic) WST'06-version of AProVE and the set of 90 termination problems for *CSR* which have been used in the 2006 termination competition:

<http://www.lri.fr/~marche/termination-competition/2006>

A summary of the benchmarks can be found here:

<http://www.dsic.upv.es/~rgutierrez/muterm/benchmarks.html>

The benchmarks were executed on a PC equipped with an AMD Athlon XP processor at 2.4 GHz and 512 MB of RAM, running Linux (kernel 2.6.12). Both AProVE

and MU-TERM succeeded (running in a completely automatic way and with a time-out of 1 minute) on 56 examples; furthermore, the total elapsed time was almost the same for both tools. The MU-TERM expert used CSDPs in 45 of the 56 cases (80.4%); CSRPO in 7 cases (12.5%), and transformations in only 4 cases (7.1%, three of them using Zantema's transformation and one of them using Giesl and Middeldorp's incomplete transformation).

## 5 Conclusions and Future work

We have presented MU-TERM, a tool for proving termination of *CSR*. The tool has been improved with the implementation of new direct techniques for proving termination of *CSR* (the context-sensitive dependency pairs and the context-sensitive recursive path orderings) and an 'expert' for automatically proving termination of *CSR*. The new features perform quite well and have been shown useful in comparison with previously implemented techniques.

Future extensions of the tool will address the problem of efficiently using negative coefficients in polynomial interpretations (see [21] for further motivation). More research is also necessary to make the use of rational coefficients in proofs of termination much more efficient.

The current implementation of CSRPO is based on an ad-hoc incremental constraint solver which could be improved in many different directions. We plan to explore the reduction of the problem to a SAT-solving format, as described in [5]. We also plan to develop algorithms to solve polynomial constraints over the reals yielding exact (but not necessarily rational) solutions.

Finally, we want to improve the generation of reports and the inclusion of new, richer formats for input systems (e.g., Conditional TRSs, Many sorted TRSs, TRSs with AC symbols, etc.).

## References

- [1] T. Arts and J. Giesl. Termination of Term Rewriting Using Dependency Pairs *Theoretical Computer Science*, 236:133-178, 2000.
- [2] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-Sensitive Dependency Pairs. In N. Garg and S. Arun-Kumar, editors *Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS'06*, LNCS 4337:297-308, Springer-Verlag, Berlin, 2006.
- [3] B. Alarcón and S. Lucas. Building .NET GUIs for Haskell applications. In *Proc. of .NET'06*, pages 57-66, University of West-Bohemia, 2006.
- [4] C. Borralleras, S. Lucas, and A. Rubio. Recursive Path Orderings can be Context-Sensitive. In A. Voronkov, editor *Proc. of 18th International Conference on Automated Deduction, CADE'02*, LNAI 2392:314-331, Springer-Verlag, Berlin, 2002.
- [5] M. Codish, V. Lagoon, and P. Stuckey. Solving Partial Order Constraints for LPO Termination. In F. Pfenning, editor, *Proc. of 17th International Conference on Rewriting Techniques and Applications, RTA'06*, LNCS 4098, 2006.
- [6] E. Contejean, C. Marché, A.-P. Tomás, and X. Urbain. Mechanically proving termination using polynomial interpretations. *Journal of Automated Reasoning*, 32(4):315-355, 2006.
- [7] N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

- [8] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Termination of Membership Equational Programs. In P. Sestoft and N. Heintze, editors, *Proc. of ACM SIGPLAN 2004 Symposium on Partial Evaluation and Program Manipulation, PEPM'04*, pages 147–158, ACM Press, New York, 2004.
- [9] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving Operational Termination of Membership Equational Programs. *Higher-Order and Symbolic Computation*, to appear, 2007.
- [10] B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In C. Kirchner, editor, *Proc. of 4th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, PPDP'02*, pages 50–61, ACM Press, New York, 2002.
- [11] B. Gramlich and S. Lucas. Simple termination of context-sensitive rewriting. In B. Fischer and E. Visser, editors, *Proc. of 3rd ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02*, pages 29–41, ACM Press, New York, 2002.
- [12] J. Giesl and A. Middeldorp. Transforming Context-Sensitive Rewrite Systems. In P. Narendran and M. Rusinowitch, editors, *Proc. of 10th International Conference on Rewriting Techniques and Applications, RTA'99*, LNCS 1631:271–285, Springer-Verlag, Berlin, 1999.
- [13] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4): 379–427, 2004.
- [14] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In F. Baader and A. Voronkov, editors *Proc. of 11th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning, LPAR'04*, LNCS 3452:301–331, Springer-Verlag, Berlin, 2004.
- [15] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In U. Furbach and N. Shankar, editors, *Proc. of Third International Joint Conference on Automated Reasoning, IJCAR'06*, LNAI 4130:281–286, Springer-Verlag, Berlin, 2006. Available at <http://www-12.informatik.rwth-aachen.de/AProVE>.
- [16] N. Hirokawa and A. Middeldorp. Tyrolean Termination Tool. In J. Giesl, editor, *Proc. of 16th International Conference on Rewriting Techniques and Applications, RTA'05*, LNCS 3467:175–184, 2005.
- [17] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, January 1998.
- [18] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):293–343, 2002.
- [19] S. Lucas. MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In V. van Oostrom, editor, *Proc. of 15th International Conference on Rewriting Techniques and Applications, RTA'04*, LNCS 3091:200–209, Springer-Verlag, Berlin, 2004.
- [20] S. Lucas. Polynomials for proving termination of context-sensitive rewriting. In I. Walukiewicz, editor, *Proc. of 7th International Conference on Foundations of Software Science and Computation Structures, FOSSACS'04*, LNCS 2987:318–332, Springer-Verlag, 2004.
- [21] S. Lucas. Polynomials over the reals in proofs of termination: from theory to practice. *RAIRO Theoretical Informatics and Applications*, 39(3):547–586, 2005.
- [22] S. Lucas. Proving termination of context-sensitive rewriting by transformation. *Information and Computation*, 204(12):1782–1846, 2006.
- [23] S. Lucas. On the relative power of polynomials with real, rational, and integer coefficients in proofs of termination of rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 17(1):49–73, 2006.
- [24] E. Ohlebusch. *Advanced Topics in Term Rewriting*. Springer-Verlag, Berlin, 2002.
- [25] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. Second Edition. University of California Press, Berkeley, 1951.
- [26] H. Zantema. Termination of Context-Sensitive Rewriting. In H. Comon, editor, *Proc. of 8th International Conference on Rewriting Techniques and Applications, RTA'97*, LNCS 1232:172–186, Springer-Verlag, Berlin, 1997.
- [27] H. Zantema. TORPA: Termination of String Rewriting Systems. *Journal of Automated Reasoning*, 34:105–39, 2006.

## 8.9 Usable Rules for Context-Sensitive Rewrite Systems

4. R. Gutiérrez, S. Lucas, and X. Urbain. **Usable Rules for Context-Sensitive Rewrite Systems**. In A. Voronkov, editor, *Proc. of XIX International Conference on Rewriting Techniques and Applications, RTA'08*, volume 5117 of *Lecture Notes in Computer Science*, pages 126–141, Hagenberg, Austria, 2008. Springer-Verlag.

## Usable Rules for Context-Sensitive Rewrite Systems\*

Raúl Gutiérrez<sup>1</sup>, Salvador Lucas<sup>1</sup>, and Xavier Urbain<sup>2</sup>

<sup>1</sup> DSIC, Universidad Politécnica de Valencia, Spain

{rgutierrez, slucas}@dsic.upv.es

<sup>2</sup> Cédric-CNAM, ENSIIE, France

urbain@ensiie.fr

**Abstract.** Recently, the dependency pairs (DP) approach has been generalized to context-sensitive rewriting (CSR). Although the *context-sensitive dependency pairs (CS-DP) approach* provides a very good basis for proving termination of CSR, the current developments basically correspond to a ten-years-old DP approach. Thus, the task of adapting all recently introduced dependency pairs techniques to get a more powerful approach becomes an important issue. In this direction, *usable rules* are one of the most interesting and powerful notions. Actually usable rule have been investigated in connection with proofs of *innermost termination* of CSR. However, the existing results apply to a quite restricted class of systems. In this paper, we introduce a notion of usable rules that can be used in proofs of termination of CSR with arbitrary systems. Our benchmarks show that the performance of the CS-DP approach is much better when such usable rules are considered in proofs of termination of CSR.

**Keywords:** Dependency pairs, term rewriting, termination.

### 1 Introduction

During the last decade, the impressive advances in techniques for proving termination of rewriting (remarkably the dependency pairs approach [6,10,13,14]) have succeeded in solving termination problems that stood out of reach for a long time. Roughly speaking, given a Term Rewriting System (TRS)  $\mathcal{R}$ , the dependency pairs associated to  $\mathcal{R}$  give rise to a new TRS  $\text{DP}(\mathcal{R})$  which (together with  $\mathcal{R}$ ) determines the so-called *dependency chains* whose finiteness characterizes termination of  $\mathcal{R}$ . The dependency pairs can be presented as a *dependency graph*, where the absence of infinite chains can be analyzed by considering the *cycles* in the graph. Basically, given a *cycle*  $\mathcal{C} \subseteq \text{DP}(\mathcal{R})$  in the dependency graph, we require  $l \succeq r$  for *all* rules in the TRS  $\mathcal{R}$ ,  $u \succeq v$  or  $u \sqsupset v$  for all dependency pairs  $u \rightarrow v \in \mathcal{C}$  and  $u \sqsupset v$  for *at least one*  $u \rightarrow v \in \mathcal{C}$ . Here,  $\succeq$  is a stable

\* Work partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02, TIN 2007-68118-C02 and HA 2006-0007.

and monotonic quasi-ordering on terms and  $\sqsupseteq$  is a *well-founded* ordering; both of them can be different for the different cycles in the dependency graph.

Termination problems with many rules require more time for getting an answer. Even worse: since termination proofs are usually constrained to succeed within a given (often short) time-out, the proof could get lost due to a lack of time. For those reasons, techniques leading to increase the efficiency (and also the power) of the dependency pairs method, like *usable rules*, appear like a key issue. Usable rules  $\mathcal{U}(\mathcal{R}, \mathcal{C}) \subseteq \mathcal{R}$  are associated to a given *cycle*  $\mathcal{C}$  of the dependency graph for  $\mathcal{R}$ . For particular (but widely used) classes of quasi-orderings  $\succeq$ , we can restrict the comparisons  $l \succeq r$  to rules  $l \rightarrow r$  in  $\mathcal{U}(\mathcal{R}, \mathcal{C})$  instead of using  $\mathcal{R}$ . Since  $\mathcal{U}(\mathcal{R}, \mathcal{C})$  is (usually) smaller than  $\mathcal{R}$ , proofs of termination often become easier in this way. Usable rules were introduced ten years ago by Arts and Giesl for proving termination of innermost rewriting [5]. The adaptation of the idea to (unrestricted) rewriting [14,17] took some years. A possible reason for that is that the proof of soundness for the innermost and for the unrestricted cases are totally different. The proof of soundness in [14,17] relies on a transformation in which all infinite (minimal) rewrite sequences can be simulated by using a *restricted* set of rules. This transformation was devised by Gramlich for a completely different purpose [15]. Later, Urbain [24] used it (with some modifications) to prove termination of rewriting modules. Finally, Hirokawa and Middeldorp [17] and (independently) Thiemann *et al.* [14] combined this idea with the idea of *usable rules* leading to an improved framework for proving termination of rewriting.

In this paper, we extend the notion of usable rule to the recently introduced dependency pairs approach for context-sensitive rewriting (CS-DPs [2,3]). Proving termination of *context-sensitive rewriting* (CSR [18,20]) is an interesting problem with many applications in the fields of term rewriting and programming languages (see [8,12,19,20,22] for further motivations). In CSR, a *replacement map* (i.e., a mapping  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  satisfying  $\mu(f) \subseteq \{1, \dots, k\}$ , for each  $k$ -ary symbol  $f$  of a signature  $\mathcal{F}$ ) is used to discriminate the argument positions on which the rewriting steps are allowed; rewriting at the topmost position is always possible. The following example gives a first intuition of CSR and CS-DPs; full details are given below.

*Example 1.* Consider the following TRS  $\mathcal{R}$  borrowed from [7, Example 4.7.37]. The program zips two lists of integers into a single one but instead of pairing the components it rather computes their quotients:

$$\text{sel}(0, \text{cons}(x, xs)) \rightarrow x \quad (1) \quad \text{sel}(s(n), \text{cons}(x, xs)) \rightarrow \text{sel}(n, xs) \quad (7)$$

$$\text{minus}(x, 0) \rightarrow x \quad (2) \quad \text{minus}(s(x), s(y)) \rightarrow \text{minus}(x, y) \quad (8)$$

$$\text{quot}(0, s(y)) \rightarrow 0 \quad (3) \quad \text{quot}(s(x), s(y)) \rightarrow s(\text{quot}(\text{minus}(x, y), s(y))) \quad (9)$$

$$\text{zWquot}(\text{nil}, x) \rightarrow \text{nil} \quad (4) \quad \text{from}(x) \rightarrow \text{cons}(x, \text{from}(s(x))) \quad (10)$$

$$\text{zWquot}(x, \text{nil}) \rightarrow \text{nil} \quad (5) \quad \text{tail}(\text{cons}(x, xs)) \rightarrow xs \quad (11)$$

$$\text{head}(\text{cons}(x, xs)) \rightarrow x \quad (6)$$

$$\text{zWquot}(\text{cons}(x, xs), \text{cons}(y, ys)) \rightarrow \text{cons}(\text{quot}(x, y), \text{zWquot}(xs, ys)) \quad (12)$$

128 R. Gutiérrez, S. Lucas, and X. Urbain

with  $\mu(\text{cons}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{ar}(f)\}$  for all other symbols  $f \in \mathcal{F}$ . The set of CS-DPs of  $\mathcal{R}$  is:

$$\begin{array}{l} \text{MINUS}(\mathfrak{s}(x), \mathfrak{s}(y)) \rightarrow \text{MINUS}(x, y) \qquad \text{SEL}(\mathfrak{s}(n), \text{cons}(x, xs)) \rightarrow \text{SEL}(n, xs) \\ \text{QUOT}(\mathfrak{s}(x), \mathfrak{s}(y)) \rightarrow \text{MINUS}(x, y) \qquad \text{ZWQUOT}(\text{cons}(x, xs), \text{cons}(y, ys)) \rightarrow \text{QUOT}(x, y) \\ \text{QUOT}(\mathfrak{s}(x), \mathfrak{s}(y)) \rightarrow \text{QUOT}(\text{minus}(x, y), \mathfrak{s}(y)) \qquad \text{SEL}(\mathfrak{s}(n), \text{cons}(x, xs)) \rightarrow xs \\ \text{TAIL}(\text{cons}(x, xs)) \rightarrow xs \end{array}$$

Note that non- $\mu$ -replacing subterms in right-hand sides (e.g.,  $\text{from}(\mathfrak{s}(x))$  in rule (10)) are *not* considered to build the CS-DPs. Also, in sharp contrast with the unrestricted case, *collapsing* dependency pairs like  $\text{TAIL}(\text{cons}(x, xs)) \rightarrow xs$  (where the right-hand side is a variable) are introduced.

Regarding proofs of termination of *innermost CSR*, the straightforward adaptation of usable rules to the context-sensitive setting only works for the so-called *conservative systems* (see [4]) where collapsing dependency pairs do not occur. In Section 3, we show that the standard adaptation does *not* work when proofs of termination of *CSR* are attempted. In Section 4, we provide a general notion of usable rules for proving termination of *CSR*. Although we follow the same proof style, our proof of soundness differs from those in [14,15,17,24] in several aspects that we clarify below. In Section 5, we prove that it is possible to use the *standard* (simpler) notion of usable rules [14,17] in proofs of termination of *CSR* for a restricted class of CS-TRSs: the *strongly conservative systems*. Section 6 provides experimental evaluations and Section 7 concludes. Complete proofs are given in [16].

## 2 Preliminaries

We assume knowledge about standard definitions and notations for term rewriting (including dependency pairs) as given in, e.g., [23]. In the following, we provide some definitions and notation on CSR [18,20] and CS-DPs [2,3].

*Context-Sensitive Rewriting.* Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$ , we consider the signature  $\mathcal{F}$  as the disjoint union  $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$  of *constructors* symbols  $c \in \mathcal{C}$  and *defined* symbols  $f \in \mathcal{D}$  where  $\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} - \mathcal{D}$ . A mapping  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  is a *replacement map* (or  $\mathcal{F}$ -map) if  $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$  [18]. Let  $M_{\mathcal{F}}$  be the set of all  $\mathcal{F}$ -maps ( $M_{\mathcal{R}}$  for the  $\mathcal{F}$ -maps of a TRS  $\mathcal{R} = (\mathcal{F}, R)$ ). A binary relation  $R$  on terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  is  $\mu$ -monotonic if  $tRs$  implies  $f(t_1, \dots, t_{i-1}, t, \dots, t_n)Rf(t_1, \dots, t_{i-1}, s, \dots, t_n)$  for all  $f \in \mathcal{F}, i \in \mu(f)$ , and  $t, s, t_1, \dots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . The set of  $\mu$ -replacing positions  $\mathcal{P}os^{\mu}(t)$  of  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is:  $\mathcal{P}os^{\mu}(t) = \{\epsilon\}$ , if  $t \in \mathcal{X}$  and  $\mathcal{P}os^{\mu}(t) = \{\epsilon\} \cup \bigcup_{i \in \mu(\text{root}(t))} i.\mathcal{P}os^{\mu}(t_i)$ , if  $t \notin \mathcal{X}$ . The set of  $\mu$ -replacing variables of  $t$  is  $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$ . The  $\mu$ -replacing subterm relation  $\triangleright_{\mu}$  is defined by  $t \triangleright_{\mu} s$  if there is  $p \in \mathcal{P}os^{\mu}(t)$  such that  $s = t|_p$ . We write  $t \triangleright_{\mu} s$  if  $t \triangleright_{\mu} s$  and  $t \neq s$ . We write

$t \triangleright_{\mu} s$  to denote that  $s$  is a non- $\mu$ -replacing strict subterm of  $t$ :  $t \triangleright_{\mu} s$  if there is  $p \in \mathcal{P}os(t) - \mathcal{P}os^{\mu}(t)$  such that  $s = t|_p$ . We say that  $f \in \mathcal{F}$  is a *hidden symbol* in  $l \rightarrow r \in R$  if there exists a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  s.t.  $r \triangleright_{\mu} t$  and  $root(t) = f$ . We say that a variable  $x$  is *migrating* in  $l \rightarrow r \in R$  if  $x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$ . In *context-sensitive rewriting* (CSR [18]), we (only) rewrite terms at  $\mu$ -replacing positions:  $t$   $\mu$ -rewrites to  $s$ , written  $t \hookrightarrow_{\mu} s$  (or  $t \hookrightarrow_{\mathcal{R}, \mu} s$ ), if  $t \xrightarrow{p} s$  and  $p \in \mathcal{P}os^{\mu}(t)$ . A TRS  $\mathcal{R}$  is  $\mu$ -terminating if  $\hookrightarrow_{\mu}$  is terminating. A term  $t$  is  $\mu$ -terminating if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \hookrightarrow_{\mu} t_2 \hookrightarrow_{\mu} \dots$ . A pair  $(\mathcal{R}, \mu)$  (or triple  $(\mathcal{F}, \mu, R)$ ) where  $\mathcal{R} = (\mathcal{F}, R)$  is a TRS and  $\mu \in M_{\mathcal{R}}$  is often called a CS-TRS. We denote  $\mathcal{H}(\mathcal{R}, \mu)$  (or just  $\mathcal{H}$ , if there is no ambiguity) the set of all hidden symbols in  $(\mathcal{R}, \mu)$ .

*Context-Sensitive Dependency Pairs.* Given a TRS  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mu \in M_{\mathcal{R}}$ , the set of context-sensitive dependency pairs (CS-DPs) is  $DP(\mathcal{R}, \mu) = DP_{\mathcal{F}}(\mathcal{R}, \mu) \cup DP_{\mathcal{X}}(\mathcal{R}, \mu)$ , where  $DP_{\mathcal{F}}(\mathcal{R}, \mu)$  and  $DP_{\mathcal{X}}(\mathcal{R}, \mu)$  are obtained as follows: let  $f(t_1, \dots, t_m) \rightarrow r \in R$  and  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  be such that  $r \triangleright_{\mu} s$ . Then (1) if  $s = g(s_1, \dots, s_n)$ , for some  $g \in \mathcal{D}$ ,  $s_1, \dots, s_n \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $l \not\triangleright_{\mu} s$ , then  $f^{\sharp}(t_1, \dots, t_m) \rightarrow g^{\sharp}(s_1, \dots, s_n) \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$ ; (2) if  $s = x \in \mathcal{V}ar^{\mu}(r) - \mathcal{V}ar^{\mu}(l)$ , then  $f^{\sharp}(t_1, \dots, t_m) \rightarrow x \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$ . Here,  $f^{\sharp}$  and  $g^{\sharp}$  are new fresh symbols (called *tuple symbols*) associated to the symbols  $f$  and  $g$  respectively. The CS-DPs in  $DP_{\mathcal{X}}(\mathcal{R}, \mu)$  are called the *collapsing* CS-DPs. Let  $\mathcal{F}^{\sharp} = \mathcal{F} \cup \{f^{\sharp} \mid f \in \mathcal{F}\}$ . We extend  $\mu \in M_{\mathcal{F}}$  into  $\mu^{\sharp} \in M_{\mathcal{F}^{\sharp}}$  by  $\mu^{\sharp}(f) = \mu^{\sharp}(f^{\sharp}) = \mu(f)$  for each  $f \in \mathcal{F}$ . As usual, for  $t = f(t_1, \dots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^{\sharp}$  to denote the *marked* term  $f^{\sharp}(t_1, \dots, t_n)$ . Let  $\mathcal{T}^{\sharp}(\mathcal{F}, \mathcal{X}) = \{t^{\sharp} \mid t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}\}$  be the set of marked terms. We will also use the set  $\mathcal{P}^{\sharp}(\mathcal{F}, \mathcal{X}) = \mathcal{T}^{\sharp}(\mathcal{F}, \mathcal{X}) \times (\mathcal{T}^{\sharp}(\mathcal{F}, \mathcal{X}) \cup \mathcal{X})$ . Given  $t = f^{\sharp}(t_1, \dots, t_k) \in \mathcal{T}^{\sharp}(\mathcal{F}, \mathcal{X})$ , we write  $t^{\natural}$  to denote the *unmarked* term  $f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . As usual, capital letters denote marked symbols in examples. A set of pairs  $\mathcal{P} \subseteq \mathcal{P}^{\sharp}(\mathcal{F}, \mathcal{X})$  is decomposed into collapsing and non-collapsing pairs ( $\mathcal{P}_{\mathcal{X}}$  and  $\mathcal{P}_{\mathcal{F}}$ , respectively):  $\mathcal{P}_{\mathcal{X}} = \{u \rightarrow v \in \mathcal{P} \mid v \in \mathcal{X}\}$  and  $\mathcal{P}_{\mathcal{F}} = \mathcal{P} - \mathcal{P}_{\mathcal{X}}$ .

Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mathcal{P} \subseteq \mathcal{P}^{\sharp}(\mathcal{F}, \mathcal{X})$  and  $\mu \in M_{\mathcal{F}}$ . An  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , for  $i \geq 1$  such that there is a substitution  $\sigma$  satisfying both:

1.  $\sigma(v_i) \hookrightarrow_{\mathcal{R}, \mu^{\sharp}}^* \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{F}}$ , and
2. if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{P}_{\mathcal{X}}$ , then there is  $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) \triangleright_{\mu} s_i$  and  $s_i^{\sharp} \hookrightarrow_{\mathcal{R}, \mu^{\sharp}}^* \sigma(u_{i+1})$ .

where  $\mathcal{V}ar(v_i) \cap \mathcal{V}ar(u_j) = \emptyset$  for all  $i \neq j$  (renaming if necessary). Let  $\mathcal{M}_{\infty, \mu}$  be the set of minimal non- $\mu$ -terminating terms. Then,  $t \in \mathcal{M}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict  $\mu$ -replacing subterm of  $t$  is terminating. We say that an  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chain is *minimal* if for all  $i \geq 1$   $\sigma(v_i)$  (whenever  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{F}}$ ),  $s_i^{\sharp}$  (whenever  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{X}}$ ) are  $\mu$ -terminating w.r.t.  $\mathcal{R}$ . A CS-TRS  $\mathcal{R} = (\mathcal{F}, \mu, R)$  is  $\mu$ -terminating if and only if there is no infinite minimal  $(\mathcal{R}, DP(\mathcal{R}, \mu), \mu^{\sharp})$ -chain. For finite CS-TRSs, the CS-DPs can be presented as a *context-sensitive dependency graph* (CS-DG); there is an arc from  $u \rightarrow v \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$  to  $u' \rightarrow$

130 R. Gutiérrez, S. Lucas, and X. Urbain

$v' \in DP(\mathcal{R}, \mu)$  if there is a substitution  $\sigma$  such that  $\sigma(v) \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u')$ ; and, there is an arc from  $u \rightarrow v \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$  to  $u' \rightarrow v' \in DP(\mathcal{R}, \mu)$  if  $root(u')^{\sharp} \in \mathcal{H}$ . We consider the *strongly connected components* in this graph. A  $\mu$ -reduction pair  $(\succeq, \sqsupset)$  consists of a stable and weakly  $\mu$ -monotonic quasi-ordering  $\succeq$ , and a stable and well-founded ordering  $\sqsupset$  satisfying  $\succeq \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succeq \subseteq \sqsupset$ . From now on, we assume that all CS-TRSs are finite.

### 3 Basic Usable Rules

Consider a set of pairs  $\mathcal{P}$  and a CS-TRS  $(\mathcal{R}, \mu)$ . Then, the set of usable rules is the smallest set of rules from  $\mathcal{R}$  which are needed to capture all the infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^{\sharp})$ -chains. The rules that are responsible for generating the chains between pairs are those rules rooted by symbols that appear in the right-hand side of the pairs below the root symbol. This concept is captured by the definition of direct dependency [14,17,24]:

**Definition 1 (Direct Dependency [14,17]).** *Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$ , we say that  $f \in \mathcal{F}$  directly depends on  $g \in \mathcal{F}$ , written  $f \triangleright_d g$ , if there is a rule  $l \rightarrow r \in R$  with  $f = root(l)$  and  $g$  occurs in  $r$ .*

The set of defined symbols in a term  $t$  is  $\mathcal{DFun}(t) = \{f \mid \exists p \in \mathcal{Pos}(t), f = root(t|_p) \in \mathcal{D}\}$ . Let  $\triangleright_d^*$  be the transitive and reflexive closure of  $\triangleright_d$ . Then, we have:

**Definition 2 (Usable Rules [14,17]).** *For a set  $\mathcal{G}$  of symbols we denote by  $\mathcal{R} \upharpoonright \mathcal{G}$  the set of rewriting rules  $l \rightarrow r \in \mathcal{R}$  with  $root(l) \in \mathcal{G}$ . The set  $\mathcal{U}(\mathcal{R}, t)$  of usable rules of a term  $t$  is defined as  $\mathcal{R} \upharpoonright \{g \mid f \triangleright_d^* g \text{ for some } f \in \mathcal{DFun}(t)\}$ . If  $\mathcal{P}$  is a set of dependency pairs then  $\mathcal{U}(\mathcal{R}, \mathcal{P}) = \bigcup_{l \rightarrow r \in \mathcal{P}} \mathcal{U}(\mathcal{R}, r)$ .*

The set  $\mathcal{U}(\mathcal{R}, \mathcal{P})$  can be used instead of  $\mathcal{R}$  when looking for a reduction pair that proves termination of  $\mathcal{R}$  [14,17]. Let us now focus on CS-TRSs.

A first attempt to give a notion of usable rules for CSR is given in [4] (basic usable rules) for proofs of *innermost* termination. The results in [4] show that the straightforward generalization of Definition 2 to *CSR* (see Definition 4 below) only applies to *conservative* CS-TRSs and cycles (of CS-DPs), that is, systems having only conservative rules [22]: a rule  $l \rightarrow r \in R$  is *conservative* if  $\mathcal{Var}^{\mu}(r) \subseteq \mathcal{Var}^{\mu}(l)$ . First, we adapt Definition 1 to the CSR setting as follows:

**Definition 3 (Basic  $\mu$ -Dependency).** *Given a CS-TRS  $(\mathcal{F}, \mu, R)$ , we say that  $f \in \mathcal{F}$  has a basic  $\mu$ -dependency on  $g \in \mathcal{F}$ , written  $f \blacktriangleright_{d, \mu} g$ , if there is  $l \rightarrow r \in R$  with  $f = root(l)$  and  $g$  occurs in  $r$  at a  $\mu$ -replacing position.*

This leads to a straightforward extension of Definition 2. The set of  $\mu$ -replacing defined symbols in a term  $t$  is  $\mathcal{DFun}^{\mu}(t) = \{f \mid \exists p \in \mathcal{Pos}^{\mu}(t), f = root(t|_p) \in \mathcal{D}\}$ . Then, we have<sup>1</sup>:

<sup>1</sup> Note that, due to the focus on innermost CSR, [4, Def. 5] slightly differs from ours.

**Definition 4 (Basic Context-Sensitive Usable Rules).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The set  $\mathcal{U}_B(\mathcal{R}, \mu, t)$  of basic context-sensitive usable rules of a term  $t$  is defined as  $\mathcal{R} \upharpoonright \{g \mid f \blacktriangleright_{d,\mu}^* g \text{ for some } f \in \mathcal{DFun}^\mu(t)\}$ , where  $\blacktriangleright_{d,\mu}^*$  is the transitive and reflexive closure of  $\blacktriangleright_{d,\mu}$ . If  $\mathcal{P} \subseteq \mathcal{P}^\sharp(\mathcal{F}, \mathcal{X})$ , then  $\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) = \bigcup_{l \rightarrow r \in \mathcal{P}} \mathcal{U}_B(\mathcal{R}, \mu^\sharp, r)$ .

*Example 2.* (Continuing Example 1) The cycles in the CS-DG are:

$$\begin{aligned} \{\text{SEL}(\mathbf{s}(n), \mathbf{cons}(x, xs)) \rightarrow \text{SEL}(n, xs)\} & \quad (C_1) \\ \{\text{MINUS}(\mathbf{s}(x), \mathbf{s}(y)) \rightarrow \text{MINUS}(x, y)\} & \quad (C_2) \\ \{\text{QUOT}(\mathbf{s}(x), \mathbf{s}(y)) \rightarrow \text{QUOT}(\mathbf{minus}(x, y), \mathbf{s}(y))\} & \quad (C_3) \end{aligned}$$

Consider the cycle  $C_3$ ; then,  $\mathcal{U}_B(\mathcal{R}, \mu^\sharp, C_3)$  contains the following rules:

$$\mathbf{minus}(x, 0) \rightarrow x \quad \mathbf{minus}(\mathbf{s}(x), \mathbf{s}(y)) \rightarrow \mathbf{minus}(x, y)$$

However, as we are going to see, and in sharp contrast with [4], Definition 4 does not lead to a correct approach for proving termination of CSR, even for conservative TRSs.

*Example 3.* Consider the TRS  $\mathcal{R} = \{\mathbf{f}(\mathbf{c}(x), x) \rightarrow \mathbf{f}(x, x), \mathbf{b} \rightarrow \mathbf{c}(\mathbf{b})\}$  [4] together with  $\mu(\mathbf{f}) = \{1, 2\}$  and  $\mu(\mathbf{c}) = \emptyset$ . Note that  $(\mathcal{R}, \mu)$  is conservative (and innermost  $\mu$ -terminating, see [4]).

We have a single cycle  $C = \{\mathbf{F}(\mathbf{c}(x), x) \rightarrow \mathbf{F}(x, x)\}$ . According to Definition 4, we have no usable rules because  $\mathbf{F}(x, x)$  contains no symbol in  $\mathcal{F}$ . We could wrongly conclude  $\mu$ -termination of  $(\mathcal{R}, \mu)$ , but we have the infinite minimal  $(\mathcal{R}, C, \mu^\sharp)$ -chain  $\mathbf{F}(\mathbf{c}(\mathbf{b}), \mathbf{b}) \rightarrow \mathbf{F}(\mathbf{b}, \mathbf{b}) \leftrightarrow \mathbf{F}(\mathbf{c}(\mathbf{b}), \mathbf{b}) \rightarrow \dots$ .

In the following, we develop a correct definition of usable rules that can be applied to arbitrary CS-TRSs.

## 4 Termination of CS-TRSs with Usable Rules

As shown in [14,17], considering the set of *usable rules* instead of all the rules suffices for proving termination of  $(\mathcal{R}, \mathcal{P})$ -chains (or  $\mathcal{P}$ -minimal sequences in [17]). In [14,17], an *interpretation* of terms as sequences of their possible reducts is used<sup>2</sup>. The definition of the transformation requires adding new fresh (list constructor) symbols  $\perp, \mathbf{g} \notin \mathcal{F}$  and the (projection) rules  $\mathbf{g}(x, y) \rightarrow x, \mathbf{g}(x, y) \rightarrow y$  (the  $\pi$ -rules). In this way, infinite minimal  $(\mathcal{R}, \mathcal{P})$ -chains can be represented as infinite  $(\mathcal{U}(\mathcal{R}, \mathcal{P}) \cup \pi, \mathcal{P})$ -chains. We recall here the interpretation definition.

**Definition 5 (Interpretation [14,17]).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mathcal{G} \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  where  $\perp$  is a new constant symbol and  $\mathbf{g}$  is a new binary symbol. The interpretation  $I_{\mathcal{G}}$  is a mapping

<sup>2</sup> This method goes back to [15].

132 R. Gutiérrez, S. Lucas, and X. Urbain

from terminating terms in  $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  defined as follows:

$$I_{\mathcal{G}}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ f(I_{\mathcal{G}}(t_1), \dots, I_{\mathcal{G}}(t_n)) & \text{if } t = f(t_1 \dots t_n) \text{ and } f \notin \mathcal{G} \\ \mathbf{g}(f(I_{\mathcal{G}}(t_1), \dots, I_{\mathcal{G}}(t_n)), t') & \text{if } t = f(t_1 \dots t_n) \text{ and } f \in \mathcal{G} \end{cases}$$

where  $t' = \text{order}(\{I_{\mathcal{G}}(u) \mid t \rightarrow_{\mathcal{R}} u\})$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{g}(t, \text{order}(T - \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

The set of symbols  $\mathcal{G} \subseteq \mathcal{F}$  in Definition 5 is intended to represent the set of ‘non-usable symbols’, i.e., symbols which do not occur in the usable rules of the considered set of pairs  $\mathcal{P}$ . In rewriting, when considering infinite minimal  $(\mathcal{R}, \mathcal{P})$ -chains, we only deal with terminating terms over  $\mathcal{R}$ . The interpretation in Definition 5 is defined only for terminating terms because non-terminating terms would yield an infinite term which, actually, does *not* belong to  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$ .

Similarly, we aim at defining a  $\mu$ -interpretation  $I_{\mathcal{G}, \mu}$  that allows us to associate an infinite  $(\mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi, \mathcal{P}, \mu^\sharp)$ -chain to each infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain. Actually, the main problem is that  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chains contain non- $\mu$ -terminating terms in non- $\mu$ -replacing positions which are potentially able to reach  $\mu$ -replacing positions: subterms at a  $\mu$ -replacing position are  $\mu$ -terminating, but we do not know anything about subterms at non- $\mu$ -replacing positions. Hence, we have to define our  $\mu$ -interpretation  $I_{\mathcal{G}, \mu}$  both on  $\mu$ -terminating and non- $\mu$ -terminating terms. In [3], we have investigated the structure of infinite  $\mu$ -rewriting sequences issued from minimal non- $\mu$ -terminating terms. Intuitively, one of the main results in [3] states that terms at non- $\mu$ -replacing positions in the right-hand side of the rules are essential to track infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chains involving collapsing CS-DPs (see [3, Proposition 3.6]). These terms, by definition, are formed by *hidden symbols*. This observation gives us the key to generalize Definition 5 properly. Following Definition 5, a  $\mu$ -terminating but non-terminating term generates an infinite list. For this reason,  $I_{\mathcal{G}}$  (as a mapping from finite into finite terms) is *not* defined for non-terminating terms.

Regarding our  $\mu$ -interpretation, if we consider the rules headed by hidden symbols as *usable*, then we are avoiding such infinite  $\mu$ -interpretations of  $\mu$ -terminating terms. A non- $\mu$ -terminating term  $t$  (below a non- $\mu$ -replacing position) is treated as if its root symbol does not belong to  $\mathcal{G}$ , because if it occurs in the  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain at a  $\mu$ -replacing position, then  $t \succeq_{\mu} s$  and  $s^\sharp$  becomes the next term in the chain. To simulate all possible derivations of the terms over  $(\mathcal{R}, \mu)$  we also need to add to the system the  $\pi$ -rules. Our new  $\mu$ -interpretation is:

**Definition 6 ( $\mu$ -Interpretation).** Let  $\mathcal{R} = (\mathcal{F}, \mu, R)$  be a CS-TRS,  $\mathcal{G} \subseteq \mathcal{F}$  be such that  $\mathcal{G} \cap \mathcal{H} = \emptyset$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  where  $\perp$  is a new constant symbol and  $\mathbf{g}$  is a new binary symbol (with  $\mu(\mathbf{g}) = \{1, 2\}$ ). The  $\mu$ -interpretation  $I_{\mathcal{G}, \mu}$  is a mapping from arbitrary terms in  $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{X})$

to terms in  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  defined as follows:

$$I_{\mathcal{G},\mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ f(I_{\mathcal{G},\mu}(t_1), \dots, I_{\mathcal{G},\mu}(t_n)) & \text{if } t = f(t_1 \dots t_n) \text{ and } f \notin \mathcal{G} \\ & \text{or } t \text{ is non-}\mu\text{-terminating} \\ \mathbf{g}(f(I_{\mathcal{G},\mu}(t_1), \dots, I_{\mathcal{G},\mu}(t_n)), t') & \text{if } t = f(t_1 \dots t_n) \text{ and } f \in \mathcal{G} \\ & \text{and } t \text{ is } \mu\text{-terminating} \end{cases}$$

where  $t' = \text{order}(\{I_{\mathcal{G},\mu}(u) \mid t \hookrightarrow_{(\mathcal{R},\mu)} u\})$   
 $\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{g}(t, \text{order}(T - \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$

The set  $\mathcal{G} \subseteq \mathcal{F}$  in Definition 6 corresponds to the set of non-usable symbols as discussed below. Now, we prove that  $I_{\mathcal{G},\mu}$  is well-defined. The most important difference (and essential in our proof) among our  $\mu$ -interpretation and all previous ones [14,15,17,24] is that  $I_{\mathcal{G},\mu}$  is well-defined both for  $\mu$ -terminating or non- $\mu$ -terminating terms.

**Lemma 1.** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and let  $\mathcal{G} \subseteq \mathcal{F} - \mathcal{H}$ . Then,  $I_{\mathcal{G},\mu}$  is well-defined.*

Now, we define an appropriate notion of direct  $\mu$ -dependency. This is not straightforward as shown in the next example.

*Example 4.* Consider the following conservative non- $\mu$ -terminating CS-TRS  $\mathcal{R} = \{\mathbf{a}(x, y) \rightarrow \mathbf{b}(x, x), \mathbf{d}(x, \mathbf{e}) \rightarrow \mathbf{a}(x, x), \mathbf{b}(x, \mathbf{c}) \rightarrow \mathbf{d}(x, x), \mathbf{c} \rightarrow \mathbf{e}\}$  with  $\mu(\mathbf{a}) = \mu(\mathbf{d}) = \{1, 2\}$ ,  $\mu(\mathbf{b}) = \{1\}$  and  $\mu(\mathbf{c}) = \mu(\mathbf{e}) = \emptyset$ . The only cycle consists of the dependency pairs  $C = \{\mathbf{A}(x, y) \rightarrow \mathbf{B}(x, x), \mathbf{D}(x, \mathbf{e}) \rightarrow \mathbf{A}(x, x), \mathbf{B}(x, \mathbf{c}) \rightarrow \mathbf{D}(x, x)\}$ .

According to Definition 4, we have no basic usable rules because the right-hand sides of the dependency pairs have no defined symbols. Since we do not consider the rule  $\mathbf{c} \rightarrow \mathbf{e}$  as usable, we would assume  $\mathcal{G} = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e}\}$ . Then, we cannot simulate the infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain  $\underline{\mathbf{A}(\mathbf{c}, \mathbf{c})} \hookrightarrow \underline{\mathbf{B}(\mathbf{c}, \mathbf{c})} \hookrightarrow \underline{\mathbf{D}(\mathbf{c}, \mathbf{c})} \hookrightarrow \underline{\mathbf{D}(\mathbf{c}, \mathbf{e})} \hookrightarrow \underline{\mathbf{A}(\mathbf{c}, \mathbf{c})} \hookrightarrow \dots$  because we have:

$$s = I_{\mathcal{G},\mu}(\mathbf{A}(\mathbf{c}, \mathbf{c})) = \underline{\mathbf{A}(g(\mathbf{c}, g(\mathbf{e}, \perp)), g(\mathbf{c}, g(\mathbf{e}, \perp)))} \hookrightarrow \mathbf{B}(g(\mathbf{c}, g(\mathbf{e}, \perp)), g(\mathbf{c}, g(\mathbf{e}, \perp))) = t$$

The interpreted term  $g(\mathbf{c}, g(\mathbf{e}, \perp))$  at the  $\mu$ -replacing position 1 of  $s$  is ‘moved’ to a non- $\mu$ -replacing position 2 of  $t$ . Hence, we cannot reduce  $t$  on the second argument of  $\mathbf{B}$  to obtain the term  $\mathbf{B}(g(\mathbf{c}, g(\mathbf{e}, \perp)), \mathbf{c})$  required for applying the next CS-DP  $(\mathbf{B}(x, \mathbf{c}) \rightarrow \mathbf{D}(x, x))$  which continues the previous  $(\mathcal{R}, \mathcal{P}, \mu)$ -chain.

In order to avoid this problem, we modify Definition 3 to take into account symbols occurring at non- $\mu$ -replacing positions in the *left-hand sides* of the rules.

**Definition 7 ( $\mu$ -Dependency).** *Given a CS-TRS  $\mathcal{R} = (\mathcal{F}, \mu, R)$ , we say that  $f \in \mathcal{F}$  directly  $\mu$ -depends on  $g \in \mathcal{F}$ , written  $f \triangleright_{d,\mu} g$ , if there is a rule  $l \rightarrow r \in R$  with  $f = \text{root}(l)$  and (1)  $g$  occurs in  $r$  at a  $\mu$ -replacing position or (2)  $g$  occurs in  $l$  at a non- $\mu$ -replacing position.*

134 R. Gutiérrez, S. Lucas, and X. Urbain

Remarkably, condition (2) in Definition 7 is not very problematic in practice because most programs are *constructor systems*, which means that no defined symbols occur below the root in the left-hand side of the rules.

Now we are ready to define our notion of usable rules. The set of *non- $\mu$ -replacing* defined symbols in a term  $t$  is  $NDFun^\mu(t) = \{f \mid \exists p \in Pos(t) \text{ and } p \notin Pos^\mu(t), f = root(t|_p) \in \mathcal{D}\}$ .

**Definition 8 (Context-Sensitive Usable Rules).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{R}}$ , and  $\mathcal{P} \subseteq \mathcal{P}^\sharp(\mathcal{F}, \mathcal{X})$ . The set  $\mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P})$  of context-sensitive usable rules for  $\mathcal{P}$  is given by  $\mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P}) = \mathcal{U}_{\mathcal{H}}(\mathcal{R}, \mu) \cup \bigcup_{l \rightarrow r \in \mathcal{P}} \mathcal{U}_E(\mathcal{R}, \mu^\sharp, l \rightarrow r)$ .

where  $\mathcal{U}_E(\mathcal{R}, \mu, l \rightarrow r) = \mathcal{R} \mid \{g \mid f \triangleright_{d, \mu}^* g \text{ for some } f \in DFun^\mu(r) \cup NDFun^\mu(l)\}$   
 $\mathcal{U}_{\mathcal{H}}(\mathcal{R}, \mu) = \mathcal{R} \mid \{g \mid f \triangleright_{d, \mu}^* g \text{ for some } f \in \mathcal{H}\}$

Note that  $\mathcal{U}_E$  extends the notion of usable rules in Definition 2, by taking into account not only dependencies with symbols on the right-hand sides of the rules, but also with some symbols in proper subterms of the left-hand sides. We call  $\mathcal{U}_E(\mathcal{R}, \mu)$  the set of *extended* usable rules. On the other hand,  $\mathcal{U}_{\mathcal{H}}$  is the set of usable rules corresponding to the hidden symbols. Now, we are ready to formulate and prove our main result in this section.

**Theorem 1.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mathcal{P} \subseteq \mathcal{P}^\sharp(\mathcal{F}, \mathcal{X})$ , and  $\mu \in M_{\mathcal{F}}$ . If there exists a  $\mu$ -reduction pair  $(\succsim, \sqsupset)$  such that  $\mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi \subseteq \succsim$ ,  $\mathcal{P} \subseteq \succsim \cup \sqsupset$ , and

1. If  $\mathcal{P}_{\mathcal{X}} = \emptyset$ , then  $\mathcal{P} \cap \sqsupset \neq \emptyset$
2. If  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , then  $\sqsupset_{\mu} \subseteq \succsim$ , and
  - (a)  $\mathcal{P} \cap \sqsupset \neq \emptyset$  and  $f(x_1, \dots, x_k) \succsim f^\sharp(x_1, \dots, x_k)$  for all  $f^\sharp$  in  $\mathcal{P}$ , or
  - (b)  $f(x_1, \dots, x_k) \sqsupset f^\sharp(x_1, \dots, x_k)$  for all  $f^\sharp$  in  $\mathcal{P}$ .

Let  $\mathcal{P}_{\sqsupset} = \{u \rightarrow v \in \mathcal{P} \mid u \sqsupset v\}$ . Then there are no infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chains whenever:

1. there are no infinite minimal  $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mu^\sharp)$ -chains in in case (1) and in case (2a).
2. there are no infinite minimal  $(\mathcal{R}, (\mathcal{P} \setminus \mathcal{P}_{\mathcal{X}}) \setminus \mathcal{P}_{\sqsupset}, \mu^\sharp)$ -chains in case (2b).

*Proof (Sketch).* By contradiction. Assume that there exists an infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain  $\mathcal{A}$  but there is no infinite minimal  $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mu^\sharp)$ -chains in case (1) and (2a), or there is no infinite minimal  $(\mathcal{R}, (\mathcal{P} \setminus \mathcal{P}_{\mathcal{X}}) \setminus \mathcal{P}_{\sqsupset}, \mu^\sharp)$ -chains in case (2b). We can assume that there is a  $\mathcal{P}' \subseteq \mathcal{P}$  such that  $\mathcal{A}$  has a tail  $\mathcal{B}$  where all pairs are used infinitely often:

$$t_1 \xrightarrow[\mathcal{R}, \mu]^* u_1 \xrightarrow{\mathcal{P}'} \circ \sqsupset_{\mu}^\sharp t_2 \xrightarrow[\mathcal{R}, \mu]^* u_2 \xrightarrow{\mathcal{P}'} \circ \sqsupset_{\mu}^\sharp \dots$$

where  $s \sqsupset_{\mu}^\sharp t$  for  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $t \in \mathcal{T}^\sharp(\mathcal{F}, \mathcal{X})$  means that  $s \sqsupset_{\mu} t^\sharp$ .

Let  $\sigma$  be a substitution, we denote by  $\sigma_{I_{\mathcal{G}, \mu}}$  the substitution that assigns to each variable  $x$  the term  $I_{\mathcal{G}, \mu}(\sigma(x))$  and let  $\mathcal{G}$  be the set of defined symbols of  $\mathcal{R} \setminus \mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P})$ . We show that after applying  $I_{\mathcal{G}, \mu}$  we get an infinite  $(\mathcal{U}(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi, \mathcal{P}', \mu^\sharp)$ -chain. All terms in the infinite chain are  $\mu$ -terminating w.r.t.  $(\mathcal{R}, \mu)$ . We proceed by induction. Let  $i \geq 1$ .

– If we consider the step  $u_i \rightarrow_{\mathcal{P}'} \circ \triangleright_{\mu}^{\sharp} t_{i+1}$ , we have two possibilities:

1. There is  $l \rightarrow r \in \mathcal{P}'_{\mathcal{F}}$ , then we get:

$$I_{\mathcal{G},\mu}(u_i) \xrightarrow{\ast}_{\pi} \sigma_{\mathcal{I}_{\mathcal{G},\mu}}(l) \rightarrow_{\mathcal{P}'_{\mathcal{F}}} \sigma_{\mathcal{I}_{\mathcal{G},\mu}}(r) = I_{\mathcal{G},\mu}(r) = I_{\mathcal{G},\mu}(t_{i+1})$$

2. There is an  $l \rightarrow x \in \mathcal{P}'_{\mathcal{X}}$ , then we get:

$$I_{\mathcal{G},\mu}(u_i) \xrightarrow{\ast}_{\pi} \sigma_{\mathcal{I}_{\mathcal{G},\mu}}(l) \rightarrow_{\mathcal{P}'_{\mathcal{X}}} \sigma_{\mathcal{I}_{\mathcal{G},\mu}}(x) = I_{\mathcal{G},\mu}(\sigma(x))$$

$$\text{and } I_{\mathcal{G},\mu}(\sigma(x)) \triangleright_{\mu} I_{\mathcal{G},\mu}(t_{i+1}^{\sharp})$$

– If we consider  $t_i \xrightarrow{\ast}_{\mathcal{R},\mu} u_i$ . We get  $I_{\mathcal{G},\mu}(t_i) \xrightarrow{\ast}_{\mathcal{U}(\mathcal{R},\mu^{\sharp},\mathcal{P}) \cup \pi} I_{\mathcal{G},\mu}(u_i)$ .

Therefore we get the infinite  $(\mathcal{U}(\mathcal{R},\mu^{\sharp},\mathcal{P}), \mathcal{P}', \mu^{\sharp})$ -chain:

$$I_{\mathcal{G},\mu}(t_1) \xrightarrow{\ast}_{\mathcal{U}(\mathcal{R},\mu^{\sharp},\mathcal{P}) \cup \pi} I_{\mathcal{G},\mu}(u_1) \rightarrow_{\mathcal{P}'} \circ \triangleright_{\mu}^{\sharp} I_{\mathcal{G},\mu}(t_2) \xrightarrow{\ast}_{\mathcal{U}(\mathcal{R},\mu^{\sharp},\mathcal{P}) \cup \pi} I_{\mathcal{G},\mu}(u_2) \rightarrow_{\mathcal{P}'} \dots$$

Using the premises of the theorem, by monotonicity and stability of  $\succsim$ , we would have that  $I_{\mathcal{G},\mu}(t_i) \succsim I_{\mathcal{G},\mu}(u_i)$  for all  $i \geq 1$ . By stability of  $\sqsupset$  (and of  $\succsim$ ), we have that  $I_{\mathcal{G},\mu}(u_i) \succsim \cup \sqsupset I_{\mathcal{G},\mu}(t_{i+1})$  for all  $i \geq 1$  and  $I_{\mathcal{G},\mu}(u_i) \sqsupset I_{\mathcal{G},\mu}(t_{i+1})$  for all  $j \in J$  for an infinite set  $J = \{j_1, \dots, j_n, \dots\}$  of natural numbers  $j_1 < j_2 < \dots < j_n < \dots$ . Now, since  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$ , we would obtain an infinite sequence consisting of infinitely many  $\sqsupset$ -steps. We obtain a contradiction to the well-foundedness of  $\sqsupset$ .  $\square$

*Remark 1.* Notice that (as expected)  $\mathcal{U}(\mathcal{R}, \mathcal{P}, \mu_{\top}) = \mathcal{U}(\mathcal{R}, \mathcal{P})$ , i.e., our usable rules for CS-TRSs  $(\mathcal{R}, \mu)$  coincide with the standard definition (see Definition 2) when  $\mu = \mu_{\top}$  is considered (here,  $\mu_{\top}(f) = \{1, \dots, ar(f)\}$  for all symbols  $f \in \mathcal{F}$ , i.e., no replacement restriction is associated to any symbol).

Thanks to Theorem 1, we do not need to make all rules in  $\mathcal{R}$  compatible with the weak component  $\succsim_{\mathcal{P}}$  of a reduction pair  $(\succsim_{\mathcal{P}}, \sqsupset_{\mathcal{P}})$  associated to a given set of pairs  $\mathcal{P}$ . We just need to consider  $\mathcal{U}(\mathcal{R}, \mu^{\sharp}, \mathcal{P})$  (together with the  $\pi$ -rules).

*Example 5.* (Continuing Examples 1 and 2) Since  $\mathcal{H} \cap \mathcal{D} = \{\mathbf{from}, \mathbf{zWquot}\}$ , we have that  $\mathcal{U}(\mathcal{R}, \mu^{\sharp}, C_1)$  is:

$$\begin{array}{ll} \mathbf{minus}(x, 0) \rightarrow x & \mathbf{minus}(\mathbf{s}(x), \mathbf{s}(y)) \rightarrow \mathbf{minus}(x, y) \\ \mathbf{quot}(0, \mathbf{s}(y)) \rightarrow 0 & \mathbf{quot}(\mathbf{s}(x), \mathbf{s}(y)) \rightarrow \mathbf{s}(\mathbf{quot}(\mathbf{minus}(x, y), \mathbf{s}(y))) \\ \mathbf{zWquot}(\mathbf{nil}, x) \rightarrow \mathbf{nil} & \mathbf{from}(x) \rightarrow \mathbf{cons}(x, \mathbf{from}(\mathbf{s}(x))) \\ \mathbf{zWquot}(x, \mathbf{nil}) \rightarrow \mathbf{nil} & \\ & \mathbf{zWquot}(\mathbf{cons}(x, xs), \mathbf{cons}(y, ys)) \rightarrow \mathbf{cons}(\mathbf{quot}(x, y), \mathbf{zWquot}(xs, ys)) \end{array}$$

According to Theorem 1, the following polynomial interpretation (computed by MU-TERM [1,21]) shows the absence of infinite  $(\mathcal{R}, C_1, \mu^{\sharp})$ -chains.

$$\begin{array}{lll} [\mathbf{s}](x) = x + 1 & [\mathbf{quot}](x, y) = x + y & [\mathbf{minus}](x, y) = 0 \\ [\mathbf{from}](x) = 0 & [\mathbf{sel}](x, y) = 0 & [\mathbf{zWquot}](x, y) = x + y \\ [\mathbf{cons}](x, y) = 0 & [0](x, y) = 0 & [\mathbf{nil}](x, y) = 1 \\ [\mathbf{SEL}](x, y) = x & & \end{array}$$

136 R. Gutiérrez, S. Lucas, and X. Urbain

Note that, if the rules for `sel` were present, we could not find a linear polynomial interpretation for solving this cycle.

*Remark 2.* When considering Definition 8 (usable rules for *CSR*) and Definition 2 (standard usable rules), one can observe that, despite the fact that *CSR* is a *restriction* of rewriting, we can obtain *more* usable rules in the context-sensitive case. Examples 3 and 4 show that this is because rules associated to hidden symbols that do *not* occur in the right-hand sides of the dependency pairs in the considered cycle can play an essential role in capturing infinite  $\mu$ -rewrite sequences. Thus, for terminating TRSs  $\mathcal{R}$ , it could be sometimes easier to find a proof of  $\mu$ -termination of the CS-TRS  $(\mathcal{R}, \mu)$  if we ignore the replacement map  $\mu$ .

## 5 Improving Usable Rules

According to the discussion in Section 3, the notion of basic usable rules is not correct even for conservative systems. Still, since  $\mathcal{U}_B(\mathcal{R}, \mu, \mathcal{P})$  is contained in (and is usually smaller than)  $\mathcal{U}(\mathcal{R}, \mu, \mathcal{P})$ , it is interesting to identify a class of CS-TRSs where basic usable rules can be safely used. Then, we consider a more restrictive kind of conservative CS-TRSs: the *strongly conservative* CS-TRSs.

**Definition 9.** Let  $\mathcal{F}$  be a signature,  $\mu \in M_{\mathcal{F}}$  and  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . We denote  $\mathcal{V}ar^{\#}(t)$  the set of variables in  $t$  occurring at non- $\mu$ -replacing positions, i.e.,  $\mathcal{V}ar^{\#}(t) = \{x \in \mathcal{V}ar(t) \mid t \triangleright_{\mu} x\}$ .

**Definition 10 (Strongly Conservative).** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . A rule  $l \rightarrow r$  is *strongly conservative* if it is conservative and  $\mathcal{V}ar^{\mu}(l) \cap \mathcal{V}ar^{\#}(l) = \mathcal{V}ar^{\mu}(r) \cap \mathcal{V}ar^{\#}(r) = \emptyset$ ; and  $\mathcal{R}$  is *strongly conservative* if all rules in  $\mathcal{R}$  are strongly conservative.

Linear CS-TRSs trivially satisfy  $\mathcal{V}ar^{\mu}(l) \cap \mathcal{V}ar^{\#}(l) = \mathcal{V}ar^{\mu}(r) \cap \mathcal{V}ar^{\#}(r) = \emptyset$ . Hence, linear conservative CS-TRSs are strongly conservative. Note that the CS-TRSs in Examples 1 and 3 are not strongly conservative.

Theorem 2 below is the other main result of this paper. It shows that basic usable rules in Definition 4 can be used to improve proofs of termination of *CSR* for strongly conservative CS-TRSs. As discussed in Section 4, if we consider minimal  $(\mathcal{R}, \mathcal{P}, \mu^{\#})$ -chains, then we deal with  $\mu$ -terminating terms w.r.t.  $(\mathcal{R}, \mu)$ . We know that any  $\mu$ -replacing subterm is  $\mu$ -terminating, but we do not know anything about non- $\mu$ -replacing subterms. However, dealing with strongly conservative CS-TRSs, we ensure that non- $\mu$ -replacing subterms cannot become  $\mu$ -replacing after  $\mu$ -rewriting(s) above them. Hence, we develop a new basic  $\mu$ -interpretation  $I'_{\mathcal{G}, \mu}$  where non- $\mu$ -replacing positions are not interpreted. In contrast to  $I'_{\mathcal{G}, \mu}$  (but closer to  $I_{\mathcal{G}}$ ) our new basic  $\mu$ -interpretation is defined now for  $\mu$ -terminating terms only.

**Definition 11 (Basic  $\mu$ -Interpretation).** Let  $(\mathcal{F}, \mu, R)$  be a CS-TRS and  $\mathcal{G} \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F}^{\#} \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  where  $\perp$  is a new constant symbol and  $\mathbf{g}$  is a new binary symbol. The basic  $\mu$ -interpretation  $I'_{\mathcal{G}, \mu}$  is

a mapping from  $\mu$ -terminating terms in  $\mathcal{T}(\mathcal{F}^\sharp, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F}^\sharp \cup \{\perp, \mathbf{g}\}, \mathcal{X})$  defined as follows:

$$I'_{\mathcal{G}, \mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ f(I'_{\mathcal{G}, \mu, f, 1}(t_1), \dots, I'_{\mathcal{G}, \mu, f, n}(t_n)) & \text{if } t = f(t_1 \dots t_n) \text{ and } f \notin \mathcal{G} \\ \mathbf{g}(f(I'_{\mathcal{G}, \mu, f, 1}(t_1), \dots, I'_{\mathcal{G}, \mu, f, n}(t_n)), t') & \text{if } t = f(t_1 \dots t_n) \text{ and } f \in \mathcal{G} \end{cases}$$

$$\text{where } I'_{\mathcal{G}, \mu, f, i}(t) = \begin{cases} I'_{\mathcal{G}, \mu}(t) & \text{if } i \in \mu(f) \\ t & \text{if } i \notin \mu(f) \end{cases}$$

$$t' = \text{order}(\{I'_{\mathcal{G}, \mu}(u) \mid t \hookrightarrow_{\mathcal{R}, \mu} u\})$$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ \mathbf{g}(t, \text{order}(T - \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

It is easy to prove that the basic  $\mu$ -interpretation is well-defined (finite) for all  $\mu$ -terminating terms.

**Lemma 2.** *For each  $\mu$ -terminating term  $t$ , the term  $I'_{\mathcal{G}, \mu}(t)$  is finite.*

For the proof of our next theorem, we need some auxiliary definitions and results.

**Definition 12.** *Let  $(\mathcal{R}, \mu)$  be a CS-TRS and  $\sigma$  be a substitution and let  $\mathcal{G} \subseteq \mathcal{F}$ . We denote by  $\sigma_{I'_{\mathcal{G}, \mu}} : \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$  a function that, given a term  $t$  replaces occurrences of  $x \in \text{Var}(t)$  at position  $p$  in  $t$  by either  $I'_{\mathcal{G}, \mu}(\sigma(x))$  if  $p \in \text{Pos}^\mu(t)$ , or  $\sigma(x)$  if  $p \notin \text{Pos}^\mu(t)$ .*

**Proposition 1.** *Let  $(\mathcal{R}, \mu)$  be a CS-TRS and  $\sigma$  be a substitution and let  $\mathcal{G} \subseteq \mathcal{F}$ . Let  $t$  be a term such that  $\text{Var}^\mu(t) \cap \text{Var}^\mu(t) = \emptyset$ . Let  $\bar{\sigma}_{I'_{\mathcal{G}, \mu}, t}$  be a substitution given by*

$$\bar{\sigma}_{I'_{\mathcal{G}, \mu}, t}(x) = \begin{cases} I'_{\mathcal{G}, \mu}(\sigma(x)) & \text{if } x \in \text{Var}^\mu(t) \\ \sigma(x) & \text{otherwise} \end{cases}$$

*Then,  $\bar{\sigma}_{I'_{\mathcal{G}, \mu}, t}(t) = \sigma_{I'_{\mathcal{G}, \mu}}(t)$ .*

The following theorem shows that we can safely consider the basic usable rules (with the  $\pi$ -rules) for proving termination of strongly conservative CS-TRSs.

**Theorem 2.** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mathcal{P} \subseteq \mathcal{P}^\sharp(\mathcal{F}, \mathcal{X})$ , and  $\mu \in M_{\mathcal{F}}$ . If  $\mathcal{P} \cup \mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P})$  is strongly conservative and there exists a  $\mu$ -reduction pair  $(\succ, \sqsupset)$  such that  $\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi \subseteq \succ$ ,  $\mathcal{P} \subseteq \succ$ , and  $\mathcal{P} \cap \sqsupset \neq \emptyset$ . Let  $\mathcal{P}_\sqsupset = \{u \rightarrow v \in \mathcal{P} \mid u \sqsupset v\}$ . Then there are no infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chains whenever there are no infinite minimal  $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_\sqsupset, \mu^\sharp)$ -chains.*

*Proof (Sketch).* By contradiction. Assume that there exists an infinite minimal  $(\mathcal{R}, \mathcal{P}, \mu^\sharp)$ -chain  $\mathcal{A}$  but there is no infinite minimal  $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_\sqsupset, \mu^\sharp)$ -chains. We can assume that there is a  $\mathcal{P}' \subseteq \mathcal{P}$  such that  $\mathcal{A}$  has a tail  $\mathcal{B}$  where all pairs are used infinitely often:

$$t_1 \hookrightarrow_{\mathcal{R}, \mu}^* u_1 \rightarrow_{\mathcal{P}'} t_2 \hookrightarrow_{\mathcal{R}, \mu}^* u_2 \rightarrow_{\mathcal{P}'} \dots$$

138 R. Gutiérrez, S. Lucas, and X. Urbain

After applying the basic  $\mu$ -interpretation  $I'_{\mathcal{G},\mu}$  we obtain an infinite  $(\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi, \mathcal{P}', \mu^\sharp)$ -chain. Since all terms in the infinite  $(\mathcal{R}, \mathcal{P}', \mu^\sharp)$ -chain are  $\mu$ -terminating w.r.t.  $(\mathcal{R}, \mu)$ , we can indeed apply the basic  $\mu$ -interpretation  $I'_{\mathcal{G},\mu}$ . Let  $i \geq 1$ .

– If we consider the pair step  $u_i \rightarrow_{\mathcal{P}'} t_{i+1}$  we can obtain the following sequence:

$$I'_{\mathcal{G},\mu}(u_i) \hookrightarrow_{\pi}^* \sigma_{I'_{\mathcal{G},\mu}}(l) \hookrightarrow_{\pi}^* \bar{\sigma}_{I'_{\mathcal{G},\mu},r}(l) \rightarrow_{\mathcal{P}'} \bar{\sigma}_{I'_{\mathcal{G},\mu},r}(r) = \sigma_{I'_{\mathcal{G},\mu}}(r) = I'_{\mathcal{G},\mu}(t_{i+1})$$

– If we consider the rewrite sequence  $t_i \hookrightarrow_{\mathcal{R},\mu}^* u_i$ . All terms in it are  $\mu$ -terminating, then we get  $I'_{\mathcal{G},\mu}(t_i) \hookrightarrow_{\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi}^* I'_{\mathcal{G},\mu}(u_i)$ .

So we obtain the infinite  $\mu$ -rewrite sequence:

$$I'_{\mathcal{G},\mu}(t_1) \hookrightarrow_{\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi}^* I'_{\mathcal{G},\mu}(u_1) \hookrightarrow_{\pi}^* \circ \rightarrow_{\mathcal{P}'} I'_{\mathcal{G},\mu}(t_2) \hookrightarrow_{\mathcal{U}_B(\mathcal{R}, \mu^\sharp, \mathcal{P}) \cup \pi}^* \dots$$

Using the premise of the theorem, it is transformed into an infinite sequence consisting of  $\succsim$  and infinitely many  $\sqsupset$  steps. Using the stability condition, this contradicts the well-foundedness of  $\sqsupset$ .  $\square$

*Example 6.* (Continuing Examples 1, 2 and 5) Cycle  $C_1$  is not strongly conservative, but cycles  $C_2$  and  $C_3$  are strongly conservative. Thus, we can use their basic usable rules. Cycle  $C_2$  has no usable rules and we can easily find a polynomial interpretation to show the absence of infinite minimal  $(\mathcal{R}, C_2, \mu^\sharp)$ -chains:

$$[\mathbf{s}](x) = x + 1 \quad [\mathbf{MINUS}](x, y) = y$$

The basic usable rules  $\mathcal{U}_B(\mathcal{R}, \mu^\sharp, C_3)$  for  $C_3$  are strongly conservative (see Example 2). The following polynomial interpretation proves the absence of infinite  $(\mathcal{R}, C_3, \mu^\sharp)$ -chains:

$$[0] = 0 \quad [\mathbf{s}](x) = x + 1 \quad [\mathbf{minus}](x, y) = x \quad [\mathbf{QUOT}](x, y) = x$$

Since we dealt with cycle  $C_1$  in Example 5,  $\mu$ -termination of  $\mathcal{R}$  is proved. Until now, no tool for proving termination of *CSR* could find a proof for this  $\mathcal{R}$  in Example 1. Thanks to the results in this paper, which have been implemented in *MU-TERM*, we can easily prove  $\mu$ -termination of  $\mathcal{R}$  now.

## 6 Experiments

The techniques described in the previous sections have been implemented as part of the tool *MU-TERM* [1,21]. In order to make clear the real contribution of the new technique to the performance of the tool, we have implemented three different versions of *MU-TERM*: (1) a basic version without any kind of usable rules, (2) a second version implementing the results about usable rules described in [4], and (3) a final version that implements the usable rules described in this paper (we do not use the notion in [4] even if the TRS is conservative and innermost equivalent). Version (2) of *MU-TERM* proves termination of *CSR* as termination

**Table 1.** Comparative among the three MU-TERM versions

Tool Version	Proved	Total Time	Average Time
<b>No Usable Rules</b>	44/90	6.11s	0.14s
<b>Innermost Usable Rules</b>	52/90	11.75s	0.23s
<b>Usable Rules</b>	64/90	8.91s	0.14s

**Table 2.** Comparative over the 44 examples

Tool Version	Proved	Total Time	Average Time
<b>No Usable Rules</b>	44/90	6.11s	0.14s
<b>Innermost Usable Rules</b>	44/90	5.03s	0.11s
<b>Usable Rules</b>	44/90	3.57s	0.08s

of innermost *CSR* when the TRS is orthogonal (see [4,11]), 37 systems, and as termination of *CSR without usable rules* in the rest of cases. In order to keep the set of experiments simple (but still meaningful), we only use linear interpretations with coefficients in  $\{0, 1\}$ . The usual practice shows that this is already quite powerful (see [9] for recent benchmarks in this sense). The benchmarks have been executed in a completely automatic way with a timeout of 1 minute on each of the 90 examples in the Context-Sensitive Rewriting subcategory of the 2007 Termination Competition<sup>3</sup>. A complete report of our experiments can be found in:

<http://www.dsic.upv.es/~rgutierrez/muterm/rta08/benchmarks.html>

Table 1 summarizes our results. Our notion of usable rules works pretty well: we are able to prove 20 more examples than without any usable rules, and 12 more than with the restricted notion in [4]. Furthermore, a comparison over the 44 examples solved by all the three versions of MU-TERM, we see that version (3) of MU-TERM is 43% faster than (1) and 27% faster than (2) (see Table 2).

## 7 Conclusions

We have investigated how *usable rules* can be used to improve termination proofs of CSR when the (context-sensitive) dependency pairs approach is used to achieve the proof. In contrast to [4], the straightforward extension of the standard notion of usable rules (called here *basic* usable rules, see Definition 4) does not work for *CSR* even for the quite restrictive class of conservative (cycles of) CS-TRSs. We have shown how to adapt the notion of usable rules for their use with arbitrary CS-TRSs (Definition 8). Theorem 1 shows that the new notion of usable rules can be used in proofs of termination of CS-TRSs. Here, although the proof uses a transformation in the very same style than [14,17], the definition of the transformation is quite different from the usual one in that it applies to

<sup>3</sup> See <http://www.lri.fr/~marche/termination-competition/2007>

140 R. Gutiérrez, S. Lucas, and X. Urbain

arbitrary terms, not only terminating ones. To our knowledge, this is the first time that Gramlich's transformation [15] is adapted and used in that way. We have also introduced the notion of strongly conservative rule and CS-TRS (Definition 10). Theorem 2 shows that basic usable rules can be used in proofs of termination involving strongly conservative cycles and rules. Although we follow the proof scheme in [14,17], a number of subtleties have to be carefully addressed before getting a correct adaptation of the proof.

We have implemented our techniques as part of the tool MU-TERM [1,21]. Our experiments show that usable rules are helpful to improve proofs of termination of *CSR*. Regarding the previous work on usable rules for innermost *CSR* [4], this paper provides a fully general definition which is not restricted to conservative systems. Actually, as we show in our experiments, our framework is more powerful in practice than trying to prove termination of *CSR* as innermost termination of *CSR* with the restricted notion of usable rules in [4]. Actually, our results provide a basis for refining the notion of usable rules in the *innermost* setting, thus hopefully allowing a generalization of the results in [4].

Finally, usable rules were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007 competition of termination tools.

## References

1. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. ENTCS 188, 105–115 (2007)
2. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 297–308. Springer, Heidelberg (2006)
3. Alarcón, B., Gutiérrez, R., Lucas, S.: Improving the Context-Sensitive Dependency Graph. ENTCS 188, 91–103 (2007)
4. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In: Konev, B., Wolter, F. (eds.) FroCos 2007. LNCS (LNAI), vol. 4720, pp. 73–87. Springer, Heidelberg (2007)
5. Arts, T., Giesl, J.: Proving Innermost Normalisation Automatically. In: Comon, H. (ed.) RTA 1997. LNCS, vol. 1232, pp. 157–171. Springer, Heidelberg (1997)
6. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236(1–2), 133–178 (2000)
7. Borralleras, C.: Ordering-Based Methods for Proving Termination Automatically. PhD thesis, Departament de Llenguatges i Sistemes Informàtics, UPC (2003)
8. Durán, F., Lucas, S., Meseguer, J., Marché, C., Urbain, X.: Proving Operational Termination of Membership Equational Programs. Higher-Order and Symbolic Computation (to appear, 2008)
9. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: SAT Solving for Termination Analysis with Polynomial Interpretations. In: Marques-Silva, J., Sakallah, K.A. (eds.) SAT 2007. LNCS, vol. 4501, pp. 340–354. Springer, Heidelberg (2007)
10. Giesl, J., Arts, T., Ohlebusch, E.: Modular Termination Proofs for Rewriting Using Dependency Pairs. Journal of Symbolic Computation 34(1), 21–58 (2002)
11. Giesl, J., Middeldorp, A.: Innermost Termination of Context-Sensitive Rewriting. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 231–244. Springer, Heidelberg (2003)

12. Giesl, J., Middeldorp, A.: Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming* 14(4), 379–427 (2004)
13. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The Dependency Pair Framework: Combining Techniques for Automated Termination Proofs. In: Baader, F., Voronkov, A. (eds.) *LPAR 2004*. LNCS (LNAI), vol. 3452, pp. 301–331. Springer, Heidelberg (2005)
14. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning* 37(3), 155–203 (2006)
15. Gramlich, B.: Generalized Sufficient Conditions for Modular Termination of Rewriting. *Applicable Algebra in Engineering, Communication and Computing* 5, 131–151 (1994)
16. Gutiérrez, R., Lucas, S., Urbain, X.: Usable Rules for Context-Sensitive Rewrite System, DSIC-II/03/08. Technical report, UPV (2008)
17. Hirokawa, N., Middeldorp, A.: Tyrolean Termination Tool: Techniques and Features. *Information and Computation* 205(4), 474–511 (2007)
18. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming* 1998(1), 1–61 (1998)
19. Lucas, S.: Termination of on-demand rewriting and termination of OBJ programs. In: *Proc. of PPDP 2001*, pp. 82–93. ACM Press, New York (2001)
20. Lucas, S.: Context-Sensitive Rewriting Strategies. *Information and Computation* 178(1), 293–343 (2002)
21. Lucas, S.: MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In: van Oostrom, V. (ed.) *RTA 2004*. LNCS, vol. 3091, pp. 200–209. Springer, Heidelberg (2004), <http://zenon.dsic.upv.es/muterm/>
22. Lucas, S.: Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation* 204(12), 1782–1846 (2006)
23. Ohlebusch, E.: *Advanced Topics in Term Rewriting*. Springer, Heidelberg (2002)
24. Urbain, X.: Modular & Incremental Automated Termination Proofs. *Journal of Automated Reasoning* 32(4), 315–355 (2004)

## 8.10 Improving Context-Sensitive Dependency Pairs

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

## Improving Context-Sensitive Dependency Pairs<sup>\*</sup>

Beatriz Alarcón<sup>1</sup>, Fabian Emmes<sup>2</sup>, Carsten Fuhs<sup>2</sup>, Jürgen Giesl<sup>2</sup>,  
Raúl Gutiérrez<sup>1</sup>, Salvador Lucas<sup>1</sup>,  
Peter Schneider-Kamp<sup>2</sup>, and René Thiemann<sup>3</sup>

<sup>1</sup> DSIC, Universidad Politécnica de Valencia, Spain

<sup>2</sup> LuFG Informatik 2, RWTH Aachen University, Germany

<sup>3</sup> Institute of Computer Science, University of Innsbruck, Austria

**Abstract.** Context-sensitive dependency pairs (CS-DPs) are currently the most powerful method for automated termination analysis of context-sensitive rewriting. However, compared to DPs for ordinary rewriting, CS-DPs suffer from two main drawbacks: (a) CS-DPs can be *collapsing*. This complicates the handling of CS-DPs and makes them less powerful in practice. (b) There does not exist a “*DP framework*” for CS-DPs which would allow one to apply them in a flexible and modular way. This paper solves drawback (a) by introducing a new definition of CS-DPs. With our definition, CS-DPs are always non-collapsing and thus, they can be handled like ordinary DPs. This allows us to solve drawback (b) as well, i.e., we extend the existing DP framework for ordinary DPs to context-sensitive rewriting. We implemented our results in the tool AProVE and successfully evaluated them on a large collection of examples.

### 1 Introduction

Context-sensitive rewriting [23,24] models evaluations in programming languages. It uses a *replacement map*  $\mu$  with  $\mu(f) \subseteq \{1, \dots, \text{arity}(f)\}$  for every function symbol  $f$  to specify the argument positions of  $f$  where rewriting may take place.

*Example 1.* Consider this context-sensitive term rewrite system (CS-TRS)

$$\begin{array}{ll}
 \text{gt}(0, y) \rightarrow \text{false} & \text{p}(0) \rightarrow 0 \\
 \text{gt}(s(x), 0) \rightarrow \text{true} & \text{p}(s(x)) \rightarrow x \\
 \text{gt}(s(x), s(y)) \rightarrow \text{gt}(x, y) & \text{minus}(x, y) \rightarrow \text{if}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x) \\
 \text{if}(\text{true}, x, y) \rightarrow x & \text{div}(0, s(y)) \rightarrow 0 \\
 \text{if}(\text{false}, x, y) \rightarrow y & \text{div}(s(x), s(y)) \rightarrow s(\text{div}(\text{minus}(x, y), s(y)))
 \end{array} \quad (1)$$

with  $\mu(\text{if}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{arity}(f)\}$  for all other symbols  $f$  to model the usual behavior of *if*: in  $\text{if}(t_1, t_2, t_3)$ , one may evaluate  $t_1$ , but not  $t_2$  or  $t_3$ . It will turn out that due to  $\mu$ , this CS-TRS is indeed terminating. In contrast, if one allows arbitrary reductions, then the TRS would be non-terminating:

<sup>\*</sup> Authors from Valencia were partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grants TIN 2007-68093-C02-02 and HA 2006-0007. B. Alarcón was partially supported by the Spanish MEC/MICINN under FPU grant AP2005-3399. R. Gutiérrez was partially supported by the Spanish MEC/MICINN, under grant TIN 2004-7943-C04-02. Authors from Aachen were supported by the DAAD under grant D/06/12785 and by the DFG under grant GI 274/5-2.

$$\text{minus}(0, 0) \rightarrow^+ \text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0) \rightarrow^+ \text{if}(\dots, \text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0), \dots) \rightarrow^+ \dots$$

There are two approaches to prove termination of context-sensitive rewriting. The first approach transforms CS-TRSs to ordinary TRSs, cf. [13,26]. But transformations often generate complicated TRSs where all termination tools fail.

Therefore, it is more promising to adapt existing termination techniques from ordinary term rewriting to the context-sensitive setting. Such adaptations were done for classical methods like RPO or polynomial orders [8,19,25]. However, much more powerful techniques like the *dependency pair* (DP) method [6] are implemented in almost all current termination tools for TRSs. But for a long time, it was not clear how to adapt the DP method to context-sensitive rewriting.

This was solved first in [1]. The corresponding implementation in the tool MU-TERM [3] outperformed all previous tools for termination of CS rewriting.

Nevertheless, the existing results on CS-DPs [1,2,4,20] still have major disadvantages compared to the DP method for ordinary rewriting, since CS-DPs can be *collapsing*. To handle such DPs, one has to impose strong requirements which make the CS-DP method quite weak and which make it difficult to extend refined termination techniques based on DPs to the CS case. In particular, the *DP framework* [14,17,21], which is the most powerful formulation of the DP method for ordinary TRSs, has not yet been adapted to the CS setting.

In this paper, we solve these problems. After presenting preliminaries in Sect. 2, we introduce a new notion of *non-collapsing* CS-DPs in Sect. 3. This new notion makes it much easier to adapt termination techniques based on DPs to context-sensitive rewriting. Therefore, Sect. 4 extends the *DP framework* to the context-sensitive setting and shows that existing methods from this framework only need minor changes to apply them to context-sensitive rewriting.

All our results are implemented in the termination prover AProVE [16]. As shown by the empirical evaluation in Sect. 5, our contributions improve the power of automated termination analysis for context-sensitive rewriting substantially.

## 2 Context-Sensitive Rewriting and CS-Dependency Pairs

See [7] and [23] for basics on term rewriting and context-sensitive rewriting, respectively. Let  $\mathcal{P}os(s)$  be the set of *positions* of a term  $s$ . For a replacement map  $\mu$ , we define the *active positions*  $\mathcal{P}os^\mu(s)$ : For  $x \in \mathcal{V}$  let  $\mathcal{P}os^\mu(x) = \{\varepsilon\}$  where  $\varepsilon$  is the root position. Moreover,  $\mathcal{P}os^\mu(f(s_1, \dots, s_n)) = \{\varepsilon\} \cup \{i p \mid i \in \mu(f), p \in \mathcal{P}os^\mu(s_i)\}$ . We say that  $s \succeq_\mu t$  holds if  $t = s|_p$  for some  $p \in \mathcal{P}os^\mu(s)$  and  $s \triangleright_\mu t$  if  $s \succeq_\mu t$  and  $s \neq t$ . Moreover,  $s \triangleright_\mu t$  if  $t = s|_p$  for some  $p \in \mathcal{P}os(s) \setminus \mathcal{P}os^\mu(s)$ . We denote the ordinary subterm relations by  $\succeq$  and  $\triangleright$ .

A *CS-TRS*  $(\mathcal{R}, \mu)$  consists of a finite TRS  $\mathcal{R}$  and a replacement map  $\mu$ . We have  $s \hookrightarrow_{\mathcal{R}, \mu} t$  iff there are  $\ell \rightarrow r \in \mathcal{R}$ ,  $p \in \mathcal{P}os^\mu(s)$ , and a substitution  $\sigma$  with  $s|_p = \sigma(\ell)$  and  $t = s[\sigma(r)]_p$ . This reduction is an *innermost* step (denoted  $\overset{i}{\hookrightarrow}_{\mathcal{R}, \mu}$ ) if all  $t$  with  $s|_p \triangleright_\mu t$  are in normal form w.r.t.  $(\mathcal{R}, \mu)$ . A term  $s$  is in *normal form* w.r.t.  $(\mathcal{R}, \mu)$  if there is no term  $t$  with  $s \hookrightarrow_{\mathcal{R}, \mu} t$ . A CS-TRS  $(\mathcal{R}, \mu)$  is *terminating* if  $\hookrightarrow_{\mathcal{R}, \mu}$  is well founded and *innermost terminating* if  $\overset{i}{\hookrightarrow}_{\mathcal{R}, \mu}$  is well founded.

638 B. Alarcón et al.

Let  $\mathcal{D} = \{\text{root}(\ell) \mid \ell \rightarrow r \in \mathcal{R}\}$  be the set of *defined symbols*. For every  $f \in \mathcal{D}$ , let  $f^\sharp$  be a fresh *tuple symbol* of same arity, where we often write “ $F$ ” instead of “ $f^\sharp$ ”. For  $t = f(t_1, \dots, t_n)$  with  $f \in \mathcal{D}$ , let  $t^\sharp = f^\sharp(t_1, \dots, t_n)$ .

**Definition 2 (CS-DPs [1]).** Let  $(\mathcal{R}, \mu)$  be a CS-TRS. If  $\ell \rightarrow r \in \mathcal{R}$ ,  $r \triangleright_\mu t$ , and  $\text{root}(t) \in \mathcal{D}$ , then  $\ell^\sharp \rightarrow t^\sharp$  is an ordinary dependency pair.<sup>1</sup> If  $\ell \rightarrow r \in \mathcal{R}$ ,  $r \triangleright_\mu x$  for a variable  $x$ , and  $\ell \not\triangleright_\mu x$ , then  $\ell^\sharp \rightarrow x$  is a collapsing DP. Let  $\text{DP}_o(\mathcal{R}, \mu)$  and  $\text{DP}_c(\mathcal{R}, \mu)$  be the sets of all ordinary resp. all collapsing DPs.

*Example 3.* For the TRS of Ex. 1, we obtain the following CS-DPs.

$$\begin{array}{ll} \text{GT}(s(x), s(y)) \rightarrow \text{GT}(x, y) & (2) \quad \text{M}(x, y) \rightarrow \text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x) & (5) \\ \text{IF}(\text{true}, x, y) \rightarrow x & (3) \quad \text{M}(x, y) \rightarrow \text{GT}(y, 0) & (6) \\ \text{IF}(\text{false}, x, y) \rightarrow y & (4) \quad \text{D}(s(x), s(y)) \rightarrow \text{D}(\text{minus}(x, y), s(y)) & (7) \\ & & \text{D}(s(x), s(y)) \rightarrow \text{M}(x, y) & (8) \end{array}$$

To prove termination, one has to show that there is no infinite *chain* of DPs. For ordinary rewriting, a sequence  $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \dots$  of DPs is a *chain* if there is a substitution  $\sigma$  such that  $t_i\sigma$  reduces to  $s_{i+1}\sigma$ .<sup>2</sup> If all  $t_i\sigma$  are terminating, then the chain is *minimal* [14,17,22]. But due to the collapsing DPs, the notion of “chains” has to be adapted when it is used with CS-DPs [1]. If  $s_i \rightarrow t_i$  is a collapsing DP (i.e., if  $t_i \in \mathcal{V}$ ), then instead of  $t_i\sigma \xrightarrow{*}_{\mathcal{R}, \mu} s_{i+1}\sigma$  (and termination of  $t_i\sigma$  for minimality), one requires that there is a term  $w_i$  with  $t_i\sigma \triangleright_\mu w_i$  and  $w_i^\sharp \xrightarrow{*}_{\mathcal{R}, \mu} s_{i+1}\sigma$ . For minimal chains,  $w_i^\sharp$  must be terminating.

*Example 4.* Ex. 1 has the chain (5), (3), (5) as  $\text{IF}(\text{gt}(s(y), 0), \text{minus}(p(x), p(s(y))), x) \xrightarrow{*}_{\mathcal{R}, \mu} \text{IF}(\text{true}, \text{minus}(p(x), p(s(y))), x) \xrightarrow{(3), \mu} \text{minus}(p(x), p(s(y)))$  and  $(\text{minus}(p(x), p(s(y))))^\sharp = \text{M}(p(x), p(s(y)))$  is an instance of the left-hand side of (5).

A CS-TRS is terminating iff there is no infinite chain [1]. As in the non-CS case, the above notion of chains can also be adapted to *innermost* rewriting. Then a CS-TRS is innermost terminating iff there is no infinite innermost chain [4].

Due to the collapsing CS-DPs (and the corresponding definition of “chains”), it is not easy to extend existing techniques for proving absence of infinite chains to CS-DPs. Therefore, we now introduce a new improved definition of CS-DPs.

### 3 Non-collapsing CS-Dependency Pairs

Ordinary DPs only consider active subterms of right-hand sides. So Rule (1) of Ex. 1 only leads to the DP (5), but not to  $\text{M}(x, y) \rightarrow \text{M}(p(x), p(y))$ . However, the inactive subterm  $\text{minus}(p(x), p(y))$  of the right-hand side of (1) may become active again when applying the rule  $\text{if}(\text{true}, x, y) \rightarrow x$ . Therefore, Def. 2 creates a collapsing DP like (3) whenever a rule  $\ell \rightarrow r$  has a *migrating variable*  $x$  with  $r \triangleright_\mu x$ , but  $\ell \not\triangleright_\mu x$ . Indeed, when instantiating the *collapse-variable*  $x$  in (3) with an instance of the “hidden term”  $\text{minus}(p(x), p(y))$ , one obtains a chain which simulates the rewrite sequence from  $\text{minus}(t_1, t_2)$  over  $\text{if}(\dots, \text{minus}(p(t_1), p(t_2)), \dots)$

<sup>1</sup> A refinement is to eliminate DPs where  $\ell \triangleright_\mu t$ , cf. [1,9].

<sup>2</sup> We always assume that different occurrences of DPs are variable-disjoint and consider substitutions whose domains may be infinite.

to  $\text{minus}(p(t_1), p(t_2))$ , cf. Ex. 4. Our main observation is that collapsing DPs are only needed for certain instantiations of the variables. One might be tempted to allow only instantiations of collapse-variables by *hidden terms*.<sup>3</sup>

**Definition 5 (Hidden Term).** Let  $(\mathcal{R}, \mu)$  be a CS-TRS. We say that  $t$  is a hidden term if  $\text{root}(t) \in \mathcal{D}$  and if there exists a rule  $\ell \rightarrow r \in \mathcal{R}$  with  $r \triangleright_{\mu} t$ .

In Ex. 1, the only hidden term is  $\text{minus}(p(x), p(y))$ . But unfortunately, only allowing instantiations of collapse-variables with hidden terms would be unsound.

*Example 6.* Consider  $\mu(g) = \{1\}$ ,  $\mu(a) = \mu(b) = \mu(f) = \mu(h) = \emptyset$  and the rules

$$\begin{array}{ll} a \rightarrow f(g(b)) & (9) \quad h(x) \rightarrow x \\ f(x) \rightarrow h(x) & b \rightarrow a \end{array}$$

The CS-TRS has the following infinite rewrite sequence:

$$a \hookrightarrow_{\mathcal{R}, \mu} f(g(b)) \hookrightarrow_{\mathcal{R}, \mu} \underline{h(g(b))} \hookrightarrow_{\mathcal{R}, \mu} g(b) \hookrightarrow_{\mathcal{R}, \mu} g(a) \hookrightarrow_{\mathcal{R}, \mu} \dots$$

We obtain the following CS-DPs according to Def. 2:

$$\begin{array}{ll} A \rightarrow F(g(b)) & H(x) \rightarrow x \\ F(x) \rightarrow H(x) & B \rightarrow A \end{array} \quad (10)$$

The only hidden term is  $b$ , obtained from Rule (9). There is also an infinite chain that corresponds to the infinite reduction above. However, here the collapse-variable  $x$  in the DP (10) must be instantiated by  $g(b)$  and not by the hidden term  $b$ , cf. the underlined part above. So if one replaced (10) by  $H(b) \rightarrow b$ , there would be no infinite chain anymore and one would falsely conclude termination.

The problem in Ex. 6 is that rewrite rules may add additional symbols like  $g$  above hidden terms. This can happen if a term  $g(t)$  occurs at an inactive position in a right-hand side and if an instantiation of  $t$  could possibly reduce to a term containing a hidden term (i.e., if  $t$  has a defined symbol or a variable at an active position). Then we call  $g(\square)$  a *hiding context*, since it can “hide” a hidden term. Moreover, the composition of hiding contexts is again a hiding context.

**Definition 7 (Hiding Context).** Let  $(\mathcal{R}, \mu)$  be a CS-TRS. The function symbol  $f$  hides position  $i$  if there is a rule  $\ell \rightarrow r \in \mathcal{R}$  with  $r \triangleright_{\mu} f(r_1, \dots, r_i, \dots, r_n)$ ,  $i \in \mu(f)$ , and  $r_i$  contains a defined symbol or a variable at an active position. A context  $C$  is hiding iff  $C = \square$  or  $C$  has the form  $f(t_1, \dots, t_{i-1}, C', t_{i+1}, \dots, t_n)$  where  $f$  hides position  $i$  and  $C'$  is a hiding context.

*Example 8.* In Ex. 6,  $g$  hides position 1 due to Rule (9). So the hiding contexts are  $\square, g(\square), g(g(\square)), \dots$ . In the TRS of Ex. 1,  $\text{minus}$  hides both positions 1 and 2 and  $p$  hides position 1 due to Rule (1). So the hiding contexts are  $\square, p(\square), \text{minus}(\square, \square), p(p(\square)), \text{minus}(\square, p(\square)), \dots$

To remove collapsing DPs  $s \rightarrow x$ , we now restrict ourselves to instantiations of  $x$  with terms of the form  $C[t]$  where  $C$  is a hiding context and  $t$  is a hidden term. So in Ex. 6, the variable  $x$  in the DP (10) should only be instantiated by  $b, g(b)$ ,

<sup>3</sup> A similar notion of *hidden symbols* was presented in [2,4], but there one only used these symbols to improve one special termination technique (the *dependency graph*).

640 B. Alarcón et al.

$g(g(b))$ , etc. To represent these infinitely many instantiations in a finite way, we replace  $s \rightarrow x$  by new *unhiding* DPs (which “unhide” hidden terms).

**Definition 9 (Improved CS-DPs).** For a CS-TRS  $(\mathcal{R}, \mu)$ , if  $\text{DP}_c(\mathcal{R}, \mu) \neq \emptyset$ , we introduce a fresh<sup>4</sup> unhiding tuple symbol  $U$  and the following unhiding DPs:

- $s \rightarrow U(x)$  for every  $s \rightarrow x \in \text{DP}_c(\mathcal{R}, \mu)$ ,
- $U(f(x_1, \dots, x_i, \dots, x_n)) \rightarrow U(x_i)$  for every function symbol  $f$  of any arity  $n$  and every  $1 \leq i \leq n$  where  $f$  hides position  $i$ , and
- $U(t) \rightarrow t^\sharp$  for every hidden term  $t$ .

Let  $\text{DP}_u(\mathcal{R}, \mu)$  be the set of all unhiding DPs (where  $\text{DP}_u(\mathcal{R}, \mu) = \emptyset$ , if  $\text{DP}_c(\mathcal{R}, \mu) = \emptyset$ ). Then the set of improved CS-DPs is  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_o(\mathcal{R}, \mu) \cup \text{DP}_u(\mathcal{R}, \mu)$ .

*Example 10.* In Ex. 6, instead of (10) we get the unhiding DPs

$$H(x) \rightarrow U(x), \quad U(g(x)) \rightarrow U(x), \quad U(b) \rightarrow B.$$

Now there is indeed an infinite chain. In Ex. 1, instead of (3) and (4), we obtain:<sup>5</sup>

$$\begin{array}{ll} \text{IF}(\text{true}, x, y) \rightarrow U(x) & (11) \quad U(\text{p}(x)) \rightarrow U(x) \quad (15) \\ \text{IF}(\text{false}, x, y) \rightarrow U(y) & (12) \quad U(\text{minus}(x, y)) \rightarrow U(x) \quad (16) \\ U(\text{minus}(\text{p}(x), \text{p}(y))) \rightarrow M(\text{p}(x), \text{p}(y)) & (13) \quad U(\text{minus}(x, y)) \rightarrow U(y) \quad (17) \\ U(\text{p}(x)) \rightarrow P(x) & (14) \end{array}$$

Clearly, the improved CS-DPs are never collapsing. Thus, now the definition of (minimal)<sup>6</sup> chains is completely analogous to the one for ordinary rewriting.

**Definition 11 (Chain).** Let  $\mathcal{P}$  and  $\mathcal{R}$  be TRSs and let  $\mu$  be a replacement map. We extend  $\mu$  to tuple symbols by defining  $\mu(f^\sharp) = \mu(f)$  for all  $f \in \mathcal{D}$  and  $\mu(U) = \emptyset$ .<sup>7</sup> A sequence of pairs  $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \dots$  from  $\mathcal{P}$  is a  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain iff there is a substitution  $\sigma$  with  $t_i \sigma \hookrightarrow_{\mathcal{R}, \mu}^* s_{i+1} \sigma$  and  $t_i \sigma$  is terminating w.r.t.  $(\mathcal{R}, \mu)$  for all  $i$ . It is an innermost  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain iff  $t_i \sigma \xrightarrow{\mathcal{R}, \mu}^* s_{i+1} \sigma$ ,  $s_i \sigma$  is in normal form, and  $t_i \sigma$  is innermost terminating w.r.t.  $(\mathcal{R}, \mu)$  for all  $i$ .

Our main theorem shows that improved CS-DPs are still sound and complete.

**Theorem 12 (Soundness and Completeness of Improved CS-DPs).** A CS-TRS  $(\mathcal{R}, \mu)$  is terminating iff there is no infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu)$ -chain and innermost terminating iff there is no infinite innermost  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu)$ -chain.

*Proof.* We only prove the theorem for “full” termination. The proof for innermost termination is very similar and can be found in [5].

<sup>4</sup> Alternatively, one could also use different  $U$ -symbols for different collapsing DPs.

<sup>5</sup> We omitted the DP  $U(\text{p}(y)) \rightarrow P(y)$  that is “identical” to (14).

<sup>6</sup> Since we only regard *minimal* chains in the following, we included the “minimality requirement” in Def. 11, i.e., we require that all  $t_i \sigma$  are (innermost) terminating. As in the DP framework for ordinary rewriting, this restriction to minimal chains is needed for several DP processors (e.g., for the reduction pair processor of Thm. 21).

<sup>7</sup> We define  $\mu(U) = \emptyset$ , since the purpose of  $U$  is only to remove context around hidden terms. But during this removal,  $U$ ’s argument should not be evaluated.

**Soundness**

$\mathcal{M}_{\infty, \mu}$  contains all *minimal non-terminating terms*:  $t \in \mathcal{M}_{\infty, \mu}$  iff  $t$  is non-terminating and every  $r$  with  $t \triangleright_{\mu} r$  terminates. A term  $u$  has the *hiding property* iff

- $u \in \mathcal{M}_{\infty, \mu}$  and
- whenever  $u \triangleright_{\mu} s \sqsubseteq_{\mu} t'$  for some terms  $s$  and  $t'$  with  $t' \in \mathcal{M}_{\infty, \mu}$ , then  $t'$  is an instance of a hidden term and  $s = C[t']$  for some hiding context  $C$ .

We first prove the following claim:

$$\text{Let } u \text{ be a term with the hiding property and let } u \xrightarrow{\mathcal{R}, \mu} v \sqsupseteq_{\mu} w \quad (18)$$

with  $w \in \mathcal{M}_{\infty, \mu}$ . Then  $w$  also has the hiding property.

Let  $w \triangleright_{\mu} s \sqsubseteq_{\mu} t'$  for some terms  $s$  and  $t'$  with  $t' \in \mathcal{M}_{\infty, \mu}$ . Clearly, this also implies  $v \triangleright_{\mu} s$ . If already  $u \triangleright s$ , then we must have  $u \triangleright_{\mu} s$  due to the minimality of  $u$ . Thus,  $t'$  is an instance of a hidden term and  $s = C[t']$  for a hiding context  $C$ , since  $u$  has the hiding property. Otherwise,  $u \not\triangleright s$ . There must be a rule  $\ell \rightarrow r \in \mathcal{R}$ , an active context  $D$  (i.e., a context where the hole is at an active position), and a substitution  $\delta$  such that  $u = D[\delta(\ell)]$  and  $v = D[\delta(r)]$ . Clearly,  $u \not\triangleright s$  implies  $\delta(\ell) \not\triangleright s$  and  $D \not\triangleright s$ . Hence,  $v \triangleright_{\mu} s$  means  $\delta(r) \triangleright_{\mu} s$ . (The root of  $s$  cannot be above  $\square$  in  $D$  since those positions would be active.) Note that  $s$  cannot be at or below a variable position of  $r$ , because this would imply  $\delta(\ell) \triangleright s$ . Thus,  $s$  is an instance of a non-variable subterm of  $r$  that is at an inactive position. So there is a  $r' \notin \mathcal{V}$  with  $r \triangleright_{\mu} r'$  and  $s = \delta(r')$ . Recall that  $s \sqsubseteq_{\mu} t'$ , i.e., there is a  $p \in \text{Pos}^{\mu}(s)$  with  $s|_p = t'$ . If  $p$  is a non-variable position of  $r'$ , then  $\delta(r')|_p = t'$  and  $r'|_p$  is a subterm with defined root at an active position (since  $t' \in \mathcal{M}_{\infty, \mu}$  implies  $\text{root}(t') \in \mathcal{D}$ ). Hence,  $r'|_p$  is a hidden term and thus,  $t'$  is an instance of a hidden term. Moreover, any instance of the context  $C' = r'[\square]_p$  is hiding. So if we define  $C$  to be  $\delta(C')$ , then  $s = \delta(r') = \delta(r')|_p = \delta(C')[t'] = C[t']$  for the hiding context  $C$ . On the contrary, if  $p$  is not a non-variable position of  $r'$ , then  $p = p_1 p_2$  where  $r'|_{p_1}$  is a variable  $x$ . Now  $t'$  is an active subterm of  $\delta(x)$  (more precisely,  $\delta(x)|_{p_2} = t'$ ). Since  $x$  also occurs in  $\ell$ , we have  $\delta(\ell) \triangleright \delta(x)$  and thus  $u \triangleright \delta(x)$ . Due to the minimality of  $u$  this implies  $u \triangleright_{\mu} \delta(x)$ . Since  $u \triangleright_{\mu} \delta(x) \sqsubseteq_{\mu} t'$ , the hiding property of  $u$  implies that  $t'$  is an instance of a hidden term and that  $\delta(x) = \overline{C}[t']$  for a hiding context  $\overline{C}$ . Note that since  $r'|_{p_1}$  is a variable, the context  $C'$  around this variable is also hiding (i.e.,  $C' = r'[\square]_{p_1}$ ). Thus, the context  $C = \delta(C')[\overline{C}]$  is hiding as well and  $s = \delta(r') = \delta(r')[\delta(x)|_{p_2}]_{p_1} = \delta(C')[\overline{C}[t']] = C[t']$ .

**Proof of Thm. 12 using Claim (18)**

If  $\mathcal{R}$  is not terminating, then there is a  $t \in \mathcal{M}_{\infty, \mu}$  that is minimal w.r.t.  $\sqsupseteq$ . So there are  $t, t_i, s_i, t'_{i+1}$  such that

$$t \xrightarrow{\varepsilon^*}_{\mathcal{R}, \mu} t_1 \xrightarrow{\varepsilon}_{\mathcal{R}} s_1 \sqsupseteq_{\mu} t'_2 \xrightarrow{\varepsilon^*}_{\mathcal{R}, \mu} t_2 \xrightarrow{\varepsilon}_{\mathcal{R}} s_2 \sqsupseteq_{\mu} t'_3 \xrightarrow{\varepsilon^*}_{\mathcal{R}, \mu} t_3 \dots \quad (19)$$

where  $t_i, t'_i \in \mathcal{M}_{\infty, \mu}$  and all proper subterms of  $t$  (also at *inactive* positions) terminate. Here, “ $\varepsilon$ ” (resp. “ $\varepsilon^*$ ”) denotes reductions at (resp. strictly below) the root.

Note that (18) implies that all  $t_i$  have the hiding property. To see this, we use induction on  $i$ . Since  $t$  trivially has the hiding property (as it has no non-terminating proper subterms) and all terms in the reduction  $t \xrightarrow{\geq \varepsilon, *}_{\mathcal{R}, \mu} t_1$  are from  $\mathcal{M}_{\infty, \mu}$  (as both  $t, t_1 \in \mathcal{M}_{\infty, \mu}$ ), we conclude that  $t_1$  also has the hiding property by applying (18) repeatedly. In the induction step, if  $t_{i-1}$  has the hiding property, then one application of (18) shows that  $t'_i$  also has the hiding property. By applying (18) repeatedly, one then also shows that  $t_i$  has the hiding property.

Now we show that  $t_i^\# \xrightarrow{+}_{\text{DP}(\mathcal{R}, \mu)} t'_{i+1}^\#$  and that all terms in the reduction  $t_i^\# \xrightarrow{+}_{\text{DP}(\mathcal{R}, \mu)} t'_{i+1}^\#$  terminate w.r.t.  $(\mathcal{R}, \mu)$ . As  $t'_{i+1}^\# \xrightarrow{\geq \varepsilon, *}_{\mathcal{R}, \mu} t_{i+1}^\#$ , we get an infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu)$ -chain.

From (19) we know that there are  $\ell_i \rightarrow r_i \in \mathcal{R}$  and  $p_i \in \text{Pos}^\mu(s_i)$  with  $t_i = \ell_i \sigma$ ,  $s_i = r_i \sigma$ , and  $s_i|_{p_i} = r_i \sigma|_{p_i} = t'_{i+1}$  for all  $i$ . First let  $p_i \in \text{Pos}(r_i)$  with  $r_i|_{p_i} \notin \mathcal{V}$ . Then  $\ell_i^\# \rightarrow (r_i|_{p_i})^\# \in \text{DP}_o(\mathcal{R}, \mu)$  and  $t_i^\# = \ell_i^\# \sigma \rightarrow_{\text{DP}_o(\mathcal{R}, \mu)} (r_i|_{p_i})^\# \sigma = t'_{i+1}^\#$ . Moreover, as  $t_i, t'_{i+1} \in \mathcal{M}_{\infty, \mu}$ , the terms  $t_i^\#$  and  $t'_{i+1}^\#$  are terminating.

Now let  $p_i$  be at or below the position of a variable  $x_i$  in  $r_i$ . By minimality of  $t_i$ ,  $x_i$  only occurs at inactive positions of  $\ell_i$ . Thus,  $\ell_i^\# \rightarrow \text{U}(x_i) \in \text{DP}_u(\mathcal{R}, \mu)$  and  $r_i = C_i[x_i]$  where  $C_i$  is an active context. Recall that  $t_i = \ell_i \sigma$  has the hiding property and that  $t_i \triangleright_\mu \sigma(x_i) \triangleright_\mu t'_{i+1}$ . Thus, we have  $\sigma(x_i) = C[t'_{i+1}]$  for a hiding context  $C$  and moreover,  $t'_{i+1}$  is an instance of a hidden term. Hence we obtain:

$$\begin{aligned} t_i^\# &= \sigma(\ell_i^\#) \\ &\xrightarrow{\text{DP}_u(\mathcal{R}, \mu)} \text{U}(\sigma(x_i)) && \text{since } \ell_i^\# \rightarrow \text{U}(x_i) \in \text{DP}_u(\mathcal{R}, \mu) \\ &= \text{U}(C[t'_{i+1}]) && \text{for a hiding context } C \\ &\xrightarrow{*}_{\text{DP}_u(\mathcal{R}, \mu)} \text{U}(t'_{i+1}) && \text{since } \text{U}(C[x]) \xrightarrow{*}_{\text{DP}_u(\mathcal{R}, \mu)} \text{U}(x) \text{ for any hiding context } C \\ &\xrightarrow{\text{DP}_u(\mathcal{R}, \mu)} t'_{i+1}^\# && \text{since } t'_{i+1} \text{ is an instance of a hidden term and} \\ &&& \text{U}(t) \xrightarrow{\text{DP}_u(\mathcal{R}, \mu)} t^\# \text{ for any instance } t \text{ of a hidden term} \end{aligned}$$

All terms in the reduction above are terminating. The reason is that again  $t_i, t'_{i+1} \in \mathcal{M}_{\infty, \mu}$  implies that  $t_i^\#$  and  $t'_{i+1}^\#$  are terminating. Moreover, all terms  $\text{U}(\dots)$  are normal forms since  $\mu(\text{U}) = \emptyset$  and since  $\text{U}$  does not occur in  $\mathcal{R}$ .

### Completeness

Let there be an infinite chain  $v_1 \rightarrow w_1, v_2 \rightarrow w_2, \dots$  of improved CS-DPs. First, let the chain have an infinite tail consisting only of DPs of the form  $\text{U}(f(x_1, \dots, x_i, \dots, x_n)) \rightarrow \text{U}(x_i)$ . Since  $\mu(\text{U}) = \emptyset$ , there are terms  $t_i$  with  $\text{U}(t_1) \xrightarrow{\varepsilon}_{\text{DP}(\mathcal{R}, \mu)} \text{U}(t_2) \xrightarrow{\varepsilon}_{\text{DP}(\mathcal{R}, \mu)} \dots$ . Hence,  $t_1 \triangleright_\mu t_2 \triangleright_\mu \dots$  which contradicts the well-foundedness of  $\triangleright_\mu$ .

Now we regard the remaining case. Here the chain has infinitely many DPs  $v \rightarrow w$  with  $v = \ell^\#$  for a rule  $\ell \rightarrow r \in \mathcal{R}$ . Let  $v_i \rightarrow w_i$  be such a DP and let  $v_j \rightarrow w_j$  with  $j > i$  be the next such DP in the chain. Let  $\sigma$  be the substitution used for the chain. We show that then  $v_i^\# \sigma \xrightarrow{*}_{\mathcal{R}, \mu} C[v_j^\# \sigma]$  for an active context  $C$ . Here,  $(f^\#(t_1, \dots, t_n))^\# = f(t_1, \dots, t_n)$  for all  $f \in \mathcal{D}$ . Doing this for all such DPs implies that there is an infinite reduction w.r.t.  $(\mathcal{R}, \mu)$ .

If  $v_i \rightarrow w_i \in \text{DP}_o(\mathcal{R}, \mu)$  then the claim is trivial, because then  $j = i + 1$  and  $v_i^\# \sigma \xrightarrow{\mathcal{R}, \mu} C[w_i^\# \sigma] \xrightarrow{*}_{\mathcal{R}, \mu} C[v_{i+1}^\# \sigma]$  for some active context  $C$ .

Otherwise,  $v_i \rightarrow w_i$  has the form  $v_i \rightarrow \text{U}(x)$ . Then  $v_i^\# \sigma \xrightarrow{\mathcal{R}, \mu} C_1[\sigma(x)]$  for an active context  $C_1$ . Moreover,  $\text{U}(\sigma(x))$  reduces to  $\text{U}(\delta(t))$  for a hidden term  $t$  and

a  $\delta$  by removing hiding contexts. Since hiding contexts are active,  $\sigma(x) = C_2[\delta(t)]$  for an active context  $C_2$ . Finally,  $t^\# \delta \xrightarrow{\varepsilon, *}_{\mathcal{R}, \mu} v_j \sigma$  and thus,  $t \delta \xrightarrow{\varepsilon, *}_{\mathcal{R}, \mu} v_j \sigma$ . By defining  $C = C_1[C_2]$ , we get  $v_i^\# \sigma \xrightarrow{+}_{\mathcal{R}, \mu} C[v_j \sigma]$ .  $\square$

## 4 CS Dependency Pair Framework

By Thm. 12, (innermost) termination of a CS-TRS is equivalent to absence of infinite (innermost) chains. For ordinary rewriting, the *DP framework* is the most recent and powerful collection of methods to prove absence of infinite chains automatically. Due to our new notion of (non-collapsing) CS-DPs, adapting the DP framework to the context-sensitive case now becomes much easier.<sup>8</sup>

In the DP framework, termination techniques operate on *DP problems* instead of TRSs. Def. 13 adapts this notion to context-sensitive rewriting.

**Definition 13 (CS-DP Problem and Processor).** A CS-DP problem is a tuple  $(\mathcal{P}, \mathcal{R}, \mu, e)$ , where  $\mathcal{P}$  and  $\mathcal{R}$  are TRSs,  $\mu$  is a replacement map, and  $e \in \{\mathbf{t}, \mathbf{i}\}$  is a flag that stands for **termination** or **innermost termination**. We also call  $(\mathcal{P}, \mathcal{R}, \mu)$ -chains “ $(\mathcal{P}, \mathcal{R}, \mu, \mathbf{t})$ -chains” and we call *innermost*  $(\mathcal{P}, \mathcal{R}, \mu)$ -chains “ $(\mathcal{P}, \mathcal{R}, \mu, \mathbf{i})$ -chains”. A CS-DP problem  $(\mathcal{P}, \mathcal{R}, \mu, e)$  is finite if there is no infinite  $(\mathcal{P}, \mathcal{R}, \mu, e)$ -chain.

A CS-DP processor is a function *Proc* that takes a CS-DP problem as input and returns a possibly empty set of CS-DP problems. The processor *Proc* is sound if a CS-DP problem  $d$  is finite whenever all problems in  $\text{Proc}(d)$  are finite.

For a CS-TRS  $(\mathcal{R}, \mu)$ , the termination proof starts with the *initial DP problem*  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu, e)$  where  $e$  depends on whether one wants to prove termination or innermost termination. Then sound DP processors are applied repeatedly. If the final processors return empty sets, then (innermost) termination is proved. Since innermost termination is usually easier to show than full termination, one should use  $e = \mathbf{i}$  whenever possible. As shown in [12], termination and innermost termination coincide for CS-TRSs  $(\mathcal{R}, \mu)$  where  $\mathcal{R}$  is *orthogonal* (i.e., left-linear and without critical pairs). So  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu, \mathbf{i})$  would be the initial DP problem for Ex. 1, even when proving full termination. In Sect. 4.1 - 4.3, we recapitulate 3 important DP processors and extend them to context-sensitive rewriting.

### 4.1 Dependency Graph Processor

The first processor decomposes a DP problem into several sub-problems. To this end, one determines which pairs can follow each other in chains by constructing a *dependency graph*. In contrast to related definitions for collapsing CS-DPs in [1,4], Def. 14 is analogous to the corresponding definition for non-CS rewriting.

**Definition 14 (CS-Dependency Graph).** For a CS-DP problem  $(\mathcal{P}, \mathcal{R}, \mu, e)$ , the nodes of the  $(\mathcal{P}, \mathcal{R}, \mu, e)$ -dependency graph are the pairs of  $\mathcal{P}$ , and there is an arc from  $v \rightarrow w$  to  $s \rightarrow t$  iff  $v \rightarrow w, s \rightarrow t$  is a  $(\mathcal{P}, \mathcal{R}, \mu, e)$ -chain.

<sup>8</sup> For this reason, we omitted the proofs in this section and refer to [5] for all proofs.



also occur active in  $v$ . Now we draw an arc from  $v \rightarrow w$  to  $s \rightarrow t$  whenever  $\text{REN}_v^\mu(\text{CAP}_v^\mu(w))$  and  $s$  unify by an mgu  $\theta$  where  $v\theta$  and  $s\theta$  are in normal form.<sup>11</sup>

It turns out that for the TRS of Ex. 1, the resulting estimated dependency graph is identical to the “real” graph in Fig. 1.

### 4.2 Reduction Pair Processor

There are several processors to simplify DP problems by applying suitable *well-founded orders* (e.g., the *reduction pair processor* [17,21], the *subterm criterion processor* [22], etc.). Due to the absence of collapsing DPs, most of these processors are now straightforward to adapt to the context-sensitive setting. In the following, we present the reduction pair processor with *usable rules*, because it is the only processor whose adaption is more challenging. (The adaption is similar to the one in [4,20] for the CS-DPs of Def. 2.)

To prove that a DP problem is finite, the reduction pair processor generates constraints which should be satisfied by a  $\mu$ -reduction pair  $(\succsim, \succ)$  [1]. Here,  $\succsim$  is a stable  $\mu$ -monotonic quasi-order,  $\succ$  is a stable well-founded order, and  $\succsim$  and  $\succ$  are compatible (i.e.,  $\succ \circ \succsim \subseteq \succ$  or  $\succ \circ \succ \subseteq \succ$ ). Here,  $\mu$ -monotonicity means that  $s_i \succsim t_i$  implies  $f(s_1, \dots, s_i, \dots, s_n) \succsim f(s_1, \dots, t_i, \dots, s_n)$  whenever  $i \in \mu(f)$ .

For a DP problem  $(\mathcal{P}, \mathcal{R}, \mu, e)$ , the generated constraints ensure that some rules in  $\mathcal{P}$  are strictly decreasing (w.r.t.  $\succ$ ) and all remaining rules in  $\mathcal{P}$  and  $\mathcal{R}$  are weakly decreasing (w.r.t.  $\succsim$ ). Requiring  $\ell \succsim r$  for all  $\ell \rightarrow r \in \mathcal{R}$  ensures that in a chain  $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \dots$  with  $t_i \sigma \xrightarrow[\mathcal{R}, \mu]{*} s_{i+1} \sigma$ , we have  $t_i \sigma \succsim s_{i+1} \sigma$  for all  $i$ . Hence, if a reduction pair satisfies the constraints, then one can delete the strictly decreasing pairs from  $\mathcal{P}$  as they cannot occur infinitely often in chains.

To improve this idea, it is desirable to require only a weak decrease of *certain* instead of *all* rules. In the non-context-sensitive setting, when proving innermost termination, it is sufficient if just the *usable rules* are weakly decreasing [6]. The same is true when proving full termination, provided that  $\succsim$  is  $\mathcal{C}_e$ -compatible, i.e.,  $c(x, y) \succsim x$  and  $c(x, y) \succsim y$  holds for a fresh function symbol  $c$  [17,22].

For a term containing a symbol  $f$ , all  $f$ -rules are *usable*. Moreover, if the  $f$ -rules are usable and  $f$  depends on  $h$  (denoted  $f \blacktriangleright_{\mathcal{R}} h$ ) then the  $h$ -rules are usable as well. Here,  $f \blacktriangleright_{\mathcal{R}} h$  if  $f = h$  or if there is a symbol  $g$  with  $g \blacktriangleright_{\mathcal{R}} h$  and  $g$  occurs in the right-hand side of an  $f$ -rule. The usable rules of a DP problem are defined to be the usable rules of the right-hand sides of the DPs.

<sup>11</sup> These estimations can be improved further by adapting existing refinements to the context-sensitive case. However, different to the non-context-sensitive case, for  $e = i$  it is not sufficient to check only for unification of  $\text{CAP}_v^\mu(w)$  and  $s$  (i.e., renaming variables with  $\text{REN}_v^\mu$  is also needed). This can be seen from the non-innermost terminating CS-TRS  $(\mathcal{R}, \mu)$  from [4, Ex. 8] with  $\mathcal{R} = \{f(\mathbf{s}(x), x) \rightarrow f(x, x), \mathbf{a} \rightarrow \mathbf{s}(\mathbf{a})\}$  and  $\mu(f) = \{1\}$ ,  $\mu(\mathbf{s}) = \emptyset$ . Clearly,  $\text{CAP}_{F(\mathbf{s}(x), x)}^\mu(F(x, x)) = F(x, x)$  does not unify with  $F(\mathbf{s}(y), y)$ . In contrast,  $\text{REN}_{F(\mathbf{s}(x), x)}^\mu(\text{CAP}_{F(\mathbf{s}(x), x)}^\mu(F(x, x))) = F(x', x)$  unifies with  $F(\mathbf{s}(y), y)$ . Thus, without using  $\text{REN}_{F(\mathbf{s}(x), x)}^\mu$  one would conclude that the dependency graph has no cycle and wrongly prove (innermost) termination.

As in [4,20], Def. 18 adapts<sup>12</sup> the concept of usable rules to the CS setting, resulting in  $\mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu)$ . But as shown in [20], for CS rewriting it is also helpful to consider an alternative definition of “dependence”  $\triangleright_{\mathcal{R}, \mu}$  where  $f$  also depends on symbols from *left-hand sides* of  $f$ -rules. Let  $\mathcal{F}^\mu(t)$  (resp.  $\mathcal{F}^\#(t)$ ) contain all function symbols occurring at active (resp. inactive) positions of a term  $t$ .

**Definition 18 (CS-Usable Rules).** Let  $Rls(f) = \{\ell \rightarrow r \in \mathcal{R} \mid \text{root}(\ell) = f\}$ . For any symbols  $f, h$  and CS-TRS  $(\mathcal{R}, \mu)$ , let  $f \blacktriangleright_{\mathcal{R}, \mu} h$  if  $f = h$  or if there is a symbol  $g$  with  $g \blacktriangleright_{\mathcal{R}, \mu} h$  and a rule  $\ell \rightarrow r \in Rls(f)$  with  $g \in \mathcal{F}^\mu(r)$ . Let  $f \triangleright_{\mathcal{R}, \mu} h$  if  $f = h$  or if there is a symbol  $g$  with  $g \triangleright_{\mathcal{R}, \mu} h$  and a rule  $\ell \rightarrow r \in Rls(f)$  with  $g \in \mathcal{F}^\#(\ell) \cup \mathcal{F}(r)$ . We define two forms of usable rules:

$$\begin{aligned} \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu) &= \bigcup_{s \rightarrow t \in \mathcal{P}, f \in \mathcal{F}^\mu(t), f \blacktriangleright_{\mathcal{R}, \mu} g} Rls(g) \\ \mathcal{U}^\triangleright(\mathcal{P}, \mathcal{R}, \mu) &= \bigcup_{s \rightarrow t \in \mathcal{P}, f \in \mathcal{F}^\#(s) \cup \mathcal{F}(t), f \triangleright_{\mathcal{R}, \mu} g} Rls(g) \cup \bigcup_{\ell \rightarrow r \in \mathcal{R}, f \in \mathcal{F}^\#(r), f \triangleright_{\mathcal{R}, \mu} g} Rls(g) \end{aligned}$$

*Example 19.* We continue Ex. 17.  $\mathcal{U}^\blacktriangleright(\mathcal{P}_1, \mathcal{R}, \mu) = \emptyset$  for  $\mathcal{P}_1 = \{(2)\}$ , since there is no defined symbol at an active position in the right-hand side  $\text{GT}(x, y)$  of (2). For  $\mathcal{P}_2 = \{(7)\}$ ,  $\mathcal{U}^\blacktriangleright(\mathcal{P}_2, \mathcal{R}, \mu)$  are the minus-, if-, and gt-rules, since minus occurs at an active position in  $D(\text{minus}(x, y), s(y))$  and minus depends on if and gt. For  $\mathcal{P}_3 = \{(5), (11)-(13), (15)-(17)\}$ ,  $\mathcal{U}^\blacktriangleright(\mathcal{P}_3, \mathcal{R}, \mu)$  are the gt- and p-rules, as gt and p are the only defined symbols at active positions of right-hand sides in  $\mathcal{P}_3$ .

In contrast, all  $\mathcal{U}^\triangleright(\mathcal{P}_i, \mathcal{R}, \mu)$  contain all rules except the div-rules, as minus and p are root symbols of hidden terms and minus depends on if and gt.

As shown in [4,20], the direct adaption of the usable rules to the context-sensitive case (i.e.,  $\mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu)$ ) can only be used for *conservative* CS-TRSs (if  $e = \mathbf{i}$ ) resp. for *strongly conservative* CS-TRSs (if  $e = \mathbf{t}$ ).<sup>13</sup> Let  $\mathcal{V}^\mu(t)$  (resp.  $\mathcal{V}^\#(t)$ ) be all variables occurring at active (resp. inactive) positions of a term  $t$ .

**Definition 20 (Conservative and Strongly Conservative).** A CS-TRS  $(\mathcal{R}, \mu)$  is conservative iff  $\mathcal{V}^\mu(r) \subseteq \mathcal{V}^\mu(\ell)$  for all rules  $\ell \rightarrow r \in \mathcal{R}$ . It is strongly conservative iff it is conservative and moreover,  $\mathcal{V}^\mu(\ell) \cap \mathcal{V}^\#(\ell) = \emptyset$  and  $\mathcal{V}^\mu(r) \cap \mathcal{V}^\#(r) = \emptyset$  for all rules  $\ell \rightarrow r \in \mathcal{R}$ .

Now we can define the reduction pair processor.

**Theorem 21 (CS-Reduction Pair Processor).** Let  $(\succsim, \succ)$  be a  $\mu$ -reduction pair. For a CS-DP Problem  $d = (\mathcal{P}, \mathcal{R}, \mu, e)$ , the result of  $\text{Proc}(d)$  is

- $\{(\mathcal{P} \setminus \succ, \mathcal{R}, \mu, e)\}$ , if  $\mathcal{P} \subseteq (\succ \cup \succsim)$  and at least one of the following holds:

<sup>12</sup> The adaptations can also be extended to refined definitions of usable rules [15,17].

<sup>13</sup> The corresponding counterexamples in [4,20] show that these restrictions are still necessary for our new notion of CS-DPs. In cases where one cannot use  $\mathcal{U}^\blacktriangleright$ , one can also attempt a termination proof where one drops the replacement map, i.e., where one regards the ordinary TRS  $\mathcal{R}$  instead of the CS-TRS  $(\mathcal{R}, \mu)$ . This may be helpful, since  $\mathcal{U}^\triangleright$  is not necessarily a subset of the non-context-sensitive usable rules, as a function symbol  $f$  also  $\triangleright$ -depends on symbols from *left-hand sides* of  $f$ -rules.

- (i)  $\mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu) \subseteq \succsim, \mathcal{P} \cup \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu)$  is strongly conservative,  $\succsim$  is  $C_e$ -compatible
- (ii)  $\mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu) \subseteq \succsim, \mathcal{P} \cup \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, \mu)$  is conservative,  $e = \mathbf{i}$
- (iii)  $\mathcal{U}^\circ(\mathcal{P}, \mathcal{R}, \mu) \subseteq \succsim, \succsim$  is  $C_e$ -compatible
- (iv)  $\mathcal{R} \subseteq \succsim$

- $\{d\}$ , otherwise.

Then Proc is sound.

*Example 22.* As  $\mathcal{U}^\blacktriangleright(\mathcal{P}_1, \mathcal{R}, \mu) = \emptyset$  and  $\mathcal{P}_1 = \{(2)\}$  is even strongly conservative, by Thm. 21 (i) or (ii) we only have to orient (2), which already works with the embedding order. So  $(\mathcal{P}_1, \mathcal{R}, \mu, \mathbf{i})$  is transformed to the empty set of DP problems.

For  $\mathcal{P}_2 = \{(7)\}$ ,  $\mathcal{U}^\blacktriangleright(\mathcal{P}_2, \mathcal{R}, \mu)$  contains the if-rules which are not conservative. Hence, we use Thm. 21 (iii) with a reduction pair based on the following max-polynomial interpretation [10]:  $[D(x, y)] = [\text{minus}(x, y)] = [p(x)] = x$ ,  $[s(x)] = x + 1$ ,  $[if(x, y, z)] = \max(y, z)$ ,  $[0] = [gt(x, y)] = [true] = [false] = 0$ . Then the DP (7) is strictly decreasing and all rules from  $\mathcal{U}^\circ(\mathcal{P}_2, \mathcal{R}, \mu)$  are weakly decreasing. Thus, the processor also transforms  $(\mathcal{P}_2, \mathcal{R}, \mu, \mathbf{i})$  to the empty set of DP problems.

Finally, we regard  $\mathcal{P}_3 = \{(5), (11)-(13), (15)-(17)\}$  where we use Thm. 21 (iii) with the interpretation  $[M(x, y)] = [\text{minus}(x, y)] = x + y + 1$ ,  $[IF(x, y, z)] = [if(x, y, z)] = \max(y, z)$ ,  $[U(x)] = [p(x)] = [s(x)] = x$ ,  $[0] = [gt(x, y)] = [true] = [false] = 0$ . Then the DPs (16) and (17) are strictly decreasing, whereas all other DPs from  $\mathcal{P}_3$  and all rules from  $\mathcal{U}^\circ(\mathcal{P}_3, \mathcal{R}, \mu)$  are weakly decreasing. So the processor results in the DP problem  $(\{(5), (11)-(13), (15)\}, \mathcal{R}, \mu, \mathbf{i})$ .

Next we apply  $[M(x, y)] = [\text{minus}(x, y)] = x + 1$ ,  $[IF(x, y, z)] = \max(y, z + 1)$ ,  $[if(x, y, z)] = \max(y, z)$ ,  $[U(x)] = [p(x)] = [s(x)] = x$ ,  $[0] = [gt(x, y)] = [true] = [false] = 0$ . Now (12) is strictly decreasing and all other remaining DPs and usable rules are weakly decreasing. Removing (12) yields  $(\{(5), (11), (13), (15)\}, \mathcal{R}, \mu, \mathbf{i})$ .

Thm. 21 (iii) and (iv) are a significant improvement over previous reduction pair processors [1,2,4,20] for the CS-DPs from Def. 2. The reason is that all previous CS-reduction pair processors require that the context-sensitive subterm relation is contained in  $\succsim$  (i.e.,  $\triangleright_\mu \subseteq \succsim$ ) whenever there are collapsing DPs. This is a very hard requirement which destroys one of the main advantages of the DP method (i.e., the possibility to filter away arbitrary arguments).<sup>14</sup> With our new non-collapsing CS-DPs, this requirement is no longer needed.

*Example 23.* If one requires  $\triangleright_\mu \subseteq \succsim$ , then the reduction pair processor would fail for Ex. 1, since then one cannot make the DP (7) strictly decreasing. The reason is that due to  $2 \in \mu(\text{minus})$ ,  $\triangleright_\mu \subseteq \succsim$  implies  $\text{minus}(x, y) \succsim y$ . So one cannot “filter away” the second argument of minus. But then a strict decrease of DP (7) together with  $\mu$ -monotonicity of  $\succsim$  implies  $D(s(x), s(s(x))) \succ D(\text{minus}(x, s(x)), s(s(x))) \succ D(s(x), s(s(x)))$ , in contradiction to the well-foundedness of  $\succ$ .

<sup>14</sup> Moreover, previous CS-reduction pair processors also require  $f(x_1, \dots, x_n) \succ f^\sharp(x_1, \dots, x_n)$  for all  $f \in \mathcal{D}$  or  $f(x_1, \dots, x_n) \succ f^\sharp(x_1, \dots, x_n)$  for all  $f \in \mathcal{D}$ . This requirement also destroys an important feature of the DP method, i.e., that tuple symbols  $f^\sharp$  can be treated independently from the original corresponding symbols  $f$ . This feature often simplifies the search for suitable reduction pairs considerably.

### 4.3 Transforming Context-Sensitive Dependency Pairs

To increase the power of the DP method, there exist several processors to transform a DP into new pairs (e.g., *narrowing*, *rewriting*, *instantiating*, or *forward instantiating* DPs [17]). We now adapt the *instantiation* processor to the context-sensitive setting. Similar adaptations can also be done for the other processors.<sup>15</sup>

The idea of this processor is the following. For a DP  $s \rightarrow t$ , we investigate which DPs  $v \rightarrow w$  can occur before  $s \rightarrow t$  in chains. To this end, we use the same estimation as for dependency graphs in Sect. 4.1, i.e., we check whether there is an mgu  $\theta$  of  $\text{REN}^\mu(\text{CAP}^\mu(w))$  and  $s$  if  $e = \mathbf{t}$  and analogously for  $e = \mathbf{i}$ .<sup>16</sup> Then we replace  $s \rightarrow t$  by the new DPs  $s\theta \rightarrow t\theta$  for all such mgu's  $\theta$ . This is sound since in any chain  $\dots, v \rightarrow w, s \rightarrow t, \dots$  where an instantiation of  $w$  reduces to an instantiation of  $s$ , one could use the new DP  $s\theta \rightarrow t\theta$  instead.

**Theorem 24 (CS-Instantiation Processor).** *Let  $\mathcal{P}' = \mathcal{P} \uplus \{s \rightarrow t\}$ . For  $d = (\mathcal{P}', \mathcal{R}, \mu, e)$ , let the result of  $\text{Proc}(d)$  be  $(\mathcal{P} \cup \overline{\mathcal{P}}, \mathcal{R}, \mu, e)$  where*

$$\begin{aligned} - \overline{\mathcal{P}} &= \{s\theta \rightarrow t\theta \mid \theta = \text{mgu}(\text{REN}^\mu(\text{CAP}^\mu(w)), s), v \rightarrow w \in \mathcal{P}'\}, \text{ if } e = \mathbf{t} \\ - \overline{\mathcal{P}} &= \{s\theta \rightarrow t\theta \mid \theta = \text{mgu}(\text{REN}_v^\mu(\text{CAP}_v^\mu(w)), s), v \rightarrow w \in \mathcal{P}', s\theta, v\theta \text{ normal}\}, \text{ if } e = \mathbf{i} \end{aligned}$$

*Then  $\text{Proc}$  is sound.*

*Example 25.* For the TRS of Ex. 1, we still had to solve the problem  $(\{(5), (11), (13), (15)\}, \mathcal{R}, \mu, \mathbf{i})$ , cf. Ex. 22. DP (11) has the variable-renamed left-hand side  $\text{IF}(\text{true}, x', y')$ . So the only DP that can occur before (11) in chains is (5) with the right-hand side  $\text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x)$ . Recall  $\text{REN}^\mu(\text{CAP}^\mu(\text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x))) = \text{IF}(z', \text{minus}(p(x), p(y)), x)$ , cf. Sect. 4.1. So the mgu is  $\theta = [z'/\text{true}, x'/\text{minus}(p(x), p(y)), y'/x]$ . Hence, we can replace (11) by

$$\text{IF}(\text{true}, \text{minus}(p(x), p(y)), x) \rightarrow \text{U}(\text{minus}(p(x), p(y))) \quad (20)$$

Here the CS variant of the instantiation processor is advantageous over the non-CS one which uses CAP instead of  $\text{CAP}^\mu$ , where CAP replaces all subterms with defined root (e.g.,  $\text{minus}(p(x), p(y))$ ) by fresh variables. So the non-CS processor would not help here as it only generates a variable-renamed copy of (11).

When re-computing the dependency graph, there is no arc from (20) to (15) as  $\mu(\text{U}) = \emptyset$ . So the DP problem is decomposed into  $(\{(15)\}, \mathcal{R}, \mu, \mathbf{i})$  (which is easily solved by the reduction pair processor) and  $(\{(5), (20), (13)\}, \mathcal{R}, \mu, \mathbf{i})$ .

Now we apply the reduction pair processor again with the following rational polynomial interpretation [11]:  $[\text{M}(x, y)] = \frac{3}{2}x + \frac{1}{2}y$ ,  $[\text{minus}(x, y)] = 2x + \frac{1}{2}y$ ,  $[\text{IF}(x, y, z)] = \frac{1}{2}x + y + \frac{1}{2}z$ ,  $[\text{if}(x, y, z)] = \frac{1}{2}x + y + z$ ,  $[\text{U}(x)] = x$ ,  $[\text{p}(x)] = [\text{gt}(x, y)] = \frac{1}{2}x$ ,  $[\text{s}(x)] = 2x + 2$ ,  $[\text{true}] = 1$ ,  $[\text{false}] = [0] = 0$ . Then (20) is strictly decreasing and can be removed, whereas all other remaining DPs and usable rules

<sup>15</sup> In the papers on CS-DPs up to now, the only existing adaption of such a processor was the straightforward adaption of the *narrowing* processor in the case  $e = \mathbf{t}$ , cf. [2]. However, this processor would not help for the TRS of Ex. 1.

<sup>16</sup> The counterexample of [4, Ex. 8] in Footnote 11 again illustrates why  $\text{REN}_v^\mu$  is also needed in the innermost case (whereas this is unnecessary for non-CS rewriting).

are weakly decreasing. A last application of the dependency graph processor then detects that there is no cycle anymore and thus, it returns the empty set of DP problems. Hence, termination of the TRS from Ex. 1 is proved. As shown in our experiments in Sect. 5, this proof can easily be performed automatically.

## 5 Experiments and Conclusion

We have developed a new notion of context-sensitive dependency pairs which improves significantly over previous notions. There are two main advantages:

(1) **Easier adaption of termination techniques to CS rewriting**

Now CS-DPs are very similar to DPs for ordinary rewriting and consequently, the existing powerful termination techniques from the DP framework can easily be adapted to context-sensitive rewriting. We have demonstrated this with some of the most popular DP processors in Sect. 4. Our adaptations subsume the existing earlier adaptations of the dependency graph [2], of the usable rules [20], and of the modifications for innermost rewriting [4], which were previously developed for the notion of CS-DPs from [1].

(2) **More powerful termination analysis for CS rewriting**

Due to the absence of collapsing CS-DPs, one does not have to impose extra restrictions anymore when extending the DP processors to CS rewriting, cf. Ex. 23. Hence, the power of termination proving is increased substantially.

To substantiate Claim (2), we performed extensive experiments. We implemented our new non-collapsing CS-DPs and all DP processors from this paper in the termination prover AProVE [16].<sup>17</sup> In contrast, the prover MU-TERM [3] uses the collapsing CS-DPs. Moreover, the processors for these CS-DPs are not formulated within the DP framework and thus, they cannot be applied in the same flexible and modular way. While MU-TERM was the most powerful tool for termination analysis of context-sensitive rewriting up to now (as demonstrated by the *International Competition of Termination Tools 2007* [27]), due to our new notion of CS-DPs, now AProVE is substantially more powerful. For instance, AProVE easily proves termination of our leading example from Ex. 1, whereas MU-TERM fails. Moreover, we tested the tools on all 90 context-sensitive TRSs from the *Termination Problem Data Base* that was used in the competition. We used a time limit of 120 seconds for each example. Then MU-TERM can prove termination of 68 examples, whereas the new version of AProVE proves termination of 78 examples (including all 68 TRSs where MU-TERM is successful).<sup>18</sup> Since 4 examples are known to be non-terminating, at most 8 more of the 90 examples could potentially be detected as terminating. So due to the results of this paper, termination proving of context-sensitive rewriting has now become

<sup>17</sup> We also used the subterm criterion and forward instantiation processors, cf. Sect. 4.

<sup>18</sup> If AProVE is restricted to use exactly the same processors as MU-TERM, then it still succeeds on 74 examples. So its superiority is indeed mainly due to the new CS-DPs which enable an easy adaption of the DP framework to the CS setting.

very powerful. To experiment with our implementation and for details, we refer to <http://aprove.informatik.rwth-aachen.de/eval/CS-DPs/>.

## References

1. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-sensitive dependency pairs. In: Arun-Kumar, S., Garg, N. (eds.) FSTTCS 2006. LNCS, vol. 4337, pp. 297–308. Springer, Heidelberg (2006)
2. Alarcón, B., Gutiérrez, R., Lucas, S.: Improving the context-sensitive dependency graph. In: Proc. PROLE 2006. ENTCS, vol. 188, pp. 91–103 (2007)
3. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving termination of context-sensitive rewriting with  $\mu$ -term. Pr. PROLE 2006. ENTCS, vol. 188, p. 105–115 (2007)
4. Alarcón, B., Lucas, S.: Termination of innermost context-sensitive rewriting using dependency pairs. In: Konev, B., Wolter, F. (eds.) FroCos 2007. LNCS, vol. 4720, pp. 73–87. Springer, Heidelberg (2007)
5. Alarcón, B., Emmes, F., Fuhs, C., Giesl, J., Gutiérrez, R., Lucas, S., Schneider-Kamp, P., Thiemann, R.: Improving context-sensitive dependency pairs. Technical Report AIB-2008-13 (2008), <http://aib.informatik.rwth-aachen.de/>
6. Arts, T., Giesl, J.: Termination of term rewriting using dependency pairs. Theoretical Computer Science 236, 133–178 (2000)
7. Baader, F., Nipkow, T.: Term Rewriting and All That, Cambridge (1998)
8. Borralleras, C., Lucas, S., Rubio, A.: Recursive path orderings can be context-sensitive. In: Voronkov, A. (ed.) CADE 2002. LNCS, vol. 2392, pp. 314–331. Springer, Heidelberg (2002)
9. Dershowitz, N.: Termination by abstraction. In: Demoen, B., Lifschitz, V. (eds.) ICLP 2004. LNCS, vol. 3132, pp. 1–18. Springer, Heidelberg (2004)
10. Fuhs, C., Giesl, J., Middeldorp, A., Schneider-Kamp, P., Thiemann, R., Zankl, H.: Maximal termination. In: Voronkov, A. (ed.) RTA 2008. LNCS, vol. 5117, pp. 110–125. Springer, Heidelberg (2008)
11. Fuhs, C., Navarro-Marsset, R., Otto, C., Giesl, J., Lucas, S., Schneider-Kamp, P.: Search techniques for rational polynomial orders. In: Autexier, S., Campbell, J., Rubio, J., Sorge, V., Suzuki, M., Wiedijk, F. (eds.) AISC 2008, Calculemus 2008, and MKM 2008. LNCS (LNAI), vol. 5144, pp. 109–124. Springer, Heidelberg (2008)
12. Giesl, J., Middeldorp, A.: Innermost termination of context-sensitive rewriting. In: Ito, M., Toyama, M. (eds.) DLT 2002. LNCS, vol. 2450, pp. 231–244. Springer, Heidelberg (2003)
13. Giesl, J., Middeldorp, A.: Transformation techniques for context-sensitive rewrite systems. Journal of Functional Programming 14(4), 379–427 (2004)
14. Giesl, J., Thiemann, R., Schneider-Kamp, P.: The dependency pair framework: Combining techniques for automated termination proofs. In: Baader, F., Voronkov, A. (eds.) LPAR 2004. LNCS, vol. 3452, pp. 301–331. Springer, Heidelberg (2005)
15. Giesl, J., Thiemann, R., Schneider-Kamp, P.: Proving and disproving termination of higher-order functions. In: Gramlich, B. (ed.) FroCos 2005. LNCS (LNAI), vol. 3717, pp. 216–231. Springer, Heidelberg (2005)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic termination proofs in the dependency pair framework. In: Furbach, U., Shankar, N. (eds.) IJCAR 2006. LNCS, vol. 4130, pp. 281–286. Springer, Heidelberg (2006)

17. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and improving dependency pairs. *Journal of Automatic Reasoning* 37(3), 155–203 (2006)
18. Gramlich, B.: Generalized sufficient conditions for modular termination of rewriting. *Appl. Algebra in Engineering, Comm. and Computing* 5, 131–151 (1994)
19. Gramlich, B., Lucas, S.: Simple termination of context-sensitive rewriting. In: *Proc. RULE 2002*, pp. 29–41. ACM Press, New York (2002)
20. Gutiérrez, R., Lucas, S., Urbain, X.: Usable rules for context-sensitive rewrite systems. In: Voronkov, A. (ed.) *RTA 2008*. LNCS, vol. 5117, pp. 126–141. Springer, Heidelberg (2008)
21. Hirokawa, N., Middeldorp, A.: Automating the dependency pair method. *Information and Computation* 199(1,2), 172–199 (2005)
22. Hirokawa, N., Middeldorp, A.: Tyrolean Termination Tool: techniques and features. *Information and Computation* 205(4), 474–511 (2007)
23. Lucas, S.: Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming* 1998(1), 1–61 (1998)
24. Lucas, S.: Context-sensitive rewriting strategies. *Inf. Comp.* 178(1), 293–343 (2002)
25. Lucas, S.: Polynomials for proving termination of context-sensitive rewriting. In: Walukiewicz, I. (ed.) *FOSSACS 2004*. LNCS, vol. 2987, pp. 318–332. Springer, Heidelberg (2004)
26. Lucas, S.: Proving termination of context-sensitive rewriting by transformation. *Information and Computation* 204(12), 1782–1846 (2006)
27. Marché, C., Zantema, H.: The termination competition. In: Baader, F. (ed.) *RTA 2007*. LNCS, vol. 4533, pp. 303–313. Springer, Heidelberg (2007)
28. Urbain, X.: Modular & incremental automated termination proofs. *Journal of Automated Reasoning* 32(4), 315–355 (2004)

## 8.11 Context-Sensitive Dependency Pairs

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



Contents lists available at ScienceDirect

Information and Computation

journal homepage: [www.elsevier.com/locate/ic](http://www.elsevier.com/locate/ic)Context-sensitive dependency pairs<sup>☆</sup>Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas<sup>\*</sup>

ELP Group, DSIC, Universidad Politécnica de Valencia, Spain

## ARTICLE INFO

Article history:  
Received 11 July 2008  
Revised 6 November 2009  
Available online 3 April 2010

Keywords:  
Dependency pairs  
Term rewriting  
Program analysis  
Termination

## ABSTRACT

Termination is one of the most interesting problems when dealing with context-sensitive rewrite systems. Although a good number of techniques for proving termination of context-sensitive rewriting (CSR) have been proposed so far, the adaptation to CSR of the *dependency pair approach*, one of the most powerful techniques for proving termination of rewriting, took some time and was possible only after introducing some new notions like *collapsing dependency pairs*, which are specific for CSR. In this paper, we develop the notion of *context-sensitive dependency pair* (CSDP) and show how to use CSDPs in proofs of termination of CSR. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for automatically proving termination of CSR.

© 2010 Elsevier Inc. All rights reserved.

## 1. Introduction

Most computational systems whose operational principle is based on reducing expressions can be described and analyzed by using notions and techniques from the abstract framework of term rewriting systems (TRSs [11,61]). Such computational systems (e.g., functional, algebraic, and equational programming languages as well as theorem provers based on rewriting techniques) often incorporate a predefined reduction strategy that is used to break down the nondeterminism that is inherent to reduction relations. Eventually, this can raise problems, as each kind of strategy only behaves properly for particular classes of programs (i.e., it is normalizing, optimal, etc.). For this reason, the designers of programming languages have developed mechanisms to give the user more flexible control of the program execution. For instance, *syntactic annotations* (which are associated to arguments of symbols) have been used in programming languages such as Clean [57], Haskell [41], Lisp [54], Maude [14], OBJ2 [23], OBJ3 [36], CafeOBJ [24], etc. to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as *strictness annotations* in order to become ‘more eager’ and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as *replacement restrictions* to become ‘more lazy’, thus (hopefully) avoiding nontermination. Termination is one of the most interesting practical problems in computation and software engineering. A program or computational system is said to be *terminating* if it does not lead to any infinite computation for any possible call or input data. Ensuring termination is often a prerequisite for essential program properties like correctness. Messages reporting (a never-ending) “processing”, “waiting for an answer”, or even “abnormal termination” (which are often raised during the execution of software applications) usually correspond to nonterminating computations arising from bugs in the program.

*Context-sensitive rewriting* (CSR [44,46]) is a restriction of rewriting that has proved useful in investigating some of the aforementioned programming languages, see e.g., [13,16,17,31,45,52]. In CSR, the restriction of the rewriting computations is

<sup>☆</sup> This work has been partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under Grants TIN 2007-68093-C02 and HA 2006-0007. Beatriz Alarcón was partially supported by the Spanish MEC/MICINN under FPU Grant AP2005-3399. Raúl Gutiérrez was partially supported by the Spanish MEC/MICINN Grant TIN 2004-7943-C04-02.

<sup>\*</sup> Corresponding author.

Email addresses: [balarbon@dsic.upv.es](mailto:balarbon@dsic.upv.es) (B. Alarcón), [rgutierrez@dsic.upv.es](mailto:rgutierrez@dsic.upv.es) (R. Gutiérrez), [slucas@dsic.upv.es](mailto:slucas@dsic.upv.es) (S. Lucas).

URLs: <http://www.dsic.upv.es/~balarbon> (B. Alarcón), <http://www.dsic.upv.es/~rgutierrez> (R. Gutiérrez), <http://www.dsic.upv.es/~slucas> (S. Lucas).

```

evenNs → cons(0, incr(oddNs))
oddNs → incr(evenNs)
incr(cons(x, xs)) → cons(s(x), incr(xs))
take(0, xs) → nil
take(s(n), cons(x, xs)) → consF(x, take(n, xs))
zip(nil, xs) → nil
zip(xs, nil) → nil
zip(cons(x, xs), cons(y, ys)) → cons(frac(x, y), zip(xs, ys))
tail(cons(x, xs)) → xs
rep2(nil) → nil
rep2(cons(x, xs)) → cons(x, cons(x, rep2(xs)))
add(0, n) → n
add(s(n), m) → s(add(n, m))
prod(0, n) → 0
prod(s(n), m) → add(m, prod(n, m))
prodFrac(frac(x, y), frac(z, t)) → frac(prod(x, z), prod(y, t))
prodOfFracS(nil) → frac(s(0), s(0))
prodOfFracS(consF(p, ps)) → prodFrac(p, prodOfFracS(ps))
halfPi(n) → prodOfFracS(take(n, zip(rep2(tail(evenNs)), tail(rep2(oddNs)))))

```

Fig. 1. Computing Wallis' approximation to  $\frac{\pi}{2}$ .

first imposed on the *arguments* of function symbols  $f$  in the signature  $\mathcal{F}$ . A *signature* is a set of function symbols  $f_1, \dots, f_n, \dots$  together with an *arity* function  $ar : \mathcal{F} \rightarrow \mathbb{N}$  that establishes the number of 'arguments' associated to each symbol. A *replacement map* is a mapping  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  that satisfies  $\mu(f) \subseteq \{1, \dots, ar(f)\}$ , for each symbol  $f$  in the signature  $\mathcal{F}$  [44]. It specifies the argument positions where rewriting is allowed. In CSR, we only rewrite  $\mu$ -replacing subterms: every term  $t$  (as a whole) is  $\mu$ -replacing by definition; and  $t_i$  (as well as all its  $\mu$ -replacing subterms) is a  $\mu$ -replacing subterm of  $f(t_1, \dots, t_k)$  if  $i \in \mu(f)$ .

**Example 1.** The TRS  $\mathcal{R}$  in Fig. 1 can be used to compute approximations to  $\frac{\pi}{2}$  by using Wallis' product:  $\frac{\pi}{2} = \lim_{n \rightarrow \infty} \frac{2}{1} \frac{2}{3} \frac{4}{5} \dots \frac{2n}{2n-1} \frac{2n}{2n+1}$ . In  $\mathcal{R}$ , function symbols 0 and s are used to represent natural numbers in Peano's notation; we also have the usual arithmetic operations addition and product. Symbols cons and nil are the standard list constructors which are then used to build (possibly infinite) lists of natural numbers like evenNs (the infinite list of even numbers) and oddNs (the infinite list of odd numbers). Function incr increases all the elements of a list in one unit through the application of s. The function zip merges a pair of lists into a list of fractions, and tail returns the elements of a list after removing the first one. The function take can be used to obtain the components of a finite approximation to  $\frac{\pi}{2}$  which we multiply with prodOfFracS. Note the explicit use of consF for building *finite* lists of fractions of natural numbers by means of take, thus ensuring that the product of their elements computed by prodOfFracS is well-defined. A call halfPi( $s^n(0)$ ) for some positive number  $n > 0$  will return the desired approximation. Since  $\mathcal{R}$  is *nonterminating* (due to the first two rules), we should be careful when choosing the rewrite steps that will be issued to obtain an approximation.

With CSR we can achieve a *terminating behaviour* for this system. Consider the replacement map  $\mu$  given by:

$$\mu(\text{cons}) = \{1\} \text{ and } \mu(f) = \{1, \dots, ar(f)\} \text{ for all } f \in \mathcal{F} - \{\text{cons}\}$$

where  $\mu(\text{cons}) = \{1\}$  disallows reductions on the *list* part of the list constructor cons, thus making a kind of *lazy evaluation* of lists possible. Furthermore, the replacement restrictions imposed by the replacement map  $\mu$  are *not* an obstacle to obtaining the desired approximations: the repeated application of context-sensitive rewriting steps to an expression halfPi( $s^n(0)$ ) will obtain (disregarding the particular choice of such steps) an expression frac( $s^p(0)$ ,  $s^q(0)$ ) representing the approximation  $\frac{p}{q}$  to  $\frac{\pi}{2}$ , which is obtained by taking the first  $n$  terms in Wallis' formula (this follows from [44, Theorem 11]).

### 1.1. Termination of context-sensitive rewriting

Several methods have been developed for proving termination of CSR under a replacement map  $\mu$  for a given TRS  $\mathcal{R}$  (i.e., for proving the  $\mu$ -termination of  $\mathcal{R}$ ). Termination of CSR is an interesting problem with several applications in the fields of term rewriting and in the analysis of programming languages [8, 16, 17, 19, 21, 31, 46, 50, 59]. The development of methods and techniques for automatically proving termination is, therefore, one of the most interesting and challenging problems

ADD(s(n), m) → ADD(n, m)	(1)
EVENNS → INCR(oddNs)	(2)
EVENNS → ODDNS	(3)
HALFPI(n) → EVENNS	(4)
HALFPI(n) → ODDNS	(5)
HALFPI(n) → PRODOFFRACS(take(n, zip(rep2(tail(evenNs)), tail(rep2(oddNs)))))	(6)
HALFPI(n) → REP2(oddNs)	(7)
HALFPI(n) → REP2(tail(evenNs))	(8)
HALFPI(n) → TAIL(evenNs)	(9)
HALFPI(n) → TAIL(rep2(oddNs))	(10)
HALFPI(n) → TAKE(n, zip(rep2(tail(evenNs)), tail(rep2(oddNs))))	(11)
HALFPI(n) → ZIP(rep2(tail(evenNs)), tail(rep2(oddNs)))	(12)
INCR(cons(x, xs)) → INCR(xs)	(13)
ODDNS → EVENNS	(14)
ODDNS → INCR(evenNs)	(15)
PROD(s(n), m) → ADD(m, prod(n, m))	(16)
PROD(s(n), m) → PROD(n, m)	(17)
PRODFRAC(frac(x, y), frac(z, t)) → PROD(x, z)	(18)
PRODFRAC(frac(x, y), frac(z, t)) → PROD(y, t)	(19)
PRODOFFRACS(consF(p, ps)) → PRODFRAC(p, prodOfFracS(ps))	(20)
PRODOFFRACS(consF(p, ps)) → PRODOFFRACS(ps)	(21)
REP2(cons(x, xs)) → REP2(xs)	(22)
TAKE(s(n), cons(x, xs)) → TAKE(n, xs)	(23)
ZIP(cons(x, xs), cons(y, ys)) → ZIP(xs, ys)	(24)

Fig. 2. Dependency pairs for the TRS in Example 1.

when dealing with CSR. Furthermore, with CSR, we can *achieve* a terminating behavior with nonterminating TRSs by pruning (all) infinite rewrite sequences as shown in Example 1. Examples of tools that are able to automatically prove termination of CSR are AProVE [32], Jambox [18], MU-TERM [2,47], and VMTL [60].

In the 90s, a number of transformations that permit termination of CSR to be treated as a standard termination problem were developed (see [31,50] for recent surveys). Polynomial orderings and the context-sensitive version of the recursive path ordering were also investigated [12,26,48,49]. In [3], we adapted the *dependency pair method* [10,25,38], which is a very powerful technique for proving termination of rewriting, to CSR. In this paper, we develop and improve the original notions in [3] to incorporate recent improvements introduced by the dependency pair framework [33,35], and we obtain a powerful and modern framework that improves the current state-of-the-art of methods that can be used to automatically prove termination of CSR. Our tool MU-TERM implements the methods and techniques described in this paper.

### 1.2. Dependency pairs for context-sensitive rewriting

A TRS  $\mathcal{R}$  is terminating if there is no infinite rewrite sequence starting from any term. With regard to proofs of termination of rewriting, the dependency pair technique focuses on the following idea: the rules that are really able to produce such infinite sequences are those rules  $l \rightarrow r$  such that  $r$  contains some *defined* symbol<sup>1</sup>  $g$ . Intuitively, we can think of these rules as representing some possible (direct or indirect) recursive calls. Such recursion paths associated to each rule  $l \rightarrow r$  are represented as new rules  $u \rightarrow v$ , where  $u = f^\sharp(l_1, \dots, l_k)$  if  $l = f(l_1, \dots, l_k)$ , and where  $v = g^\sharp(s_1, \dots, s_m)$  if  $s = g(s_1, \dots, s_m)$  is a subterm of  $r$  and  $g$  is a defined symbol. The notation  $f^\sharp$  for a given symbol  $f$  means that  $f$  is *marked*. In

<sup>1</sup> A symbol  $g \in \mathcal{F}$  is defined in  $\mathcal{R}$  if there is a rule in  $\mathcal{R}$  whose left-hand side is of the form  $g(l_1, \dots, l_k)$ .

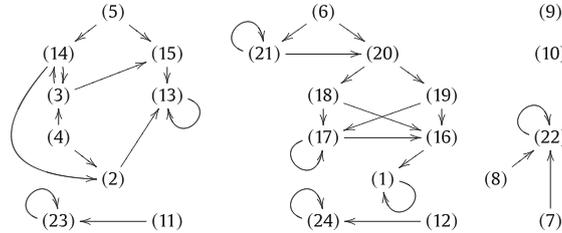


Fig. 3. Dependency graph for the TRS in Example 1.

practice, we often capitalize  $f$  and use  $F$  instead of  $f^{\sharp}$  in our examples. For this reason, the dependency pair technique starts by considering a new TRS  $\text{DP}(\mathcal{R})$  that contains all these new rules for each  $l \rightarrow r \in \mathcal{R}$ . For instance, according to [10], the set  $\text{DP}(\mathcal{R})$  of dependency pairs for  $\mathcal{R}$  in Example 1 consists of the rules in Fig. 2. The rules in  $\mathcal{R}$  and the rules in  $\text{DP}(\mathcal{R})$  determine the so-called *dependency chains* whose finiteness or infiniteness characterize termination or nontermination of  $\mathcal{R}$  [10]. A chain of dependency pairs is a sequence  $u_i \rightarrow v_i$  of dependency pairs together with a substitution  $\sigma$  such that  $\sigma(v_i)$  rewrites to  $\sigma(u_{i+1})$  for all  $i \geq 1$ . The dependency pairs can be presented as a *dependency graph*, where the infinite chains are represented by the *cycles* in the graph. For instance, the dependency graph that corresponds to the TRS  $\mathcal{R}$  in Example 1 is depicted in Fig. 3. The cycle consisting of nodes (3) and (14) witnesses the *nontermination* of  $\mathcal{R}$ .

In general, these intuitions are valid for *CSR*: the subterms  $s$  of the right-hand sides  $r$  of the rules  $l \rightarrow r$  which are considered to build the *context-sensitive dependency pairs*  $l^{\sharp} \rightarrow s^{\sharp}$  must be  $\mu$ -replacing terms now.

**Example 2.** Consider  $\mathcal{R}$  and  $\mu$  as in Example 1. Only the dependency pairs (1), (4)-(12), (14)-(21), and (23) in Fig. 2 are also context-sensitive dependency pairs.

The following example shows the need for dependency pairs of a *new kind*.

**Example 3.** Consider the following TRS  $\mathcal{R}$ :

$$a \rightarrow c(\mathcal{E}(a)) \qquad \mathcal{E}(c(x)) \rightarrow x$$

together with  $\mu(c) = \emptyset$  and  $\mu(\mathcal{E}) = \{1\}$ . No  $\mu$ -replacing subterm  $s$  in the right-hand sides of the rules is rooted by a defined symbol. Thus, there is no 'regular' dependency pair (in particular  $a \rightarrow a$  is *dismissed* due to  $\mu(c) = \emptyset$ ). If no other dependency pair is considered, we could wrongly conclude that  $\mathcal{R}$  is  $\mu$ -terminating, which is not true:

$$\mathcal{E}(a) \hookrightarrow_{\mu} \mathcal{E}(c(\mathcal{E}(a))) \hookrightarrow_{\mu} \mathcal{E}(a) \hookrightarrow_{\mu} \dots$$

Indeed, we must add the following *collapsing* dependency pair:

$$\mathcal{E}(c(x)) \rightarrow x.$$

Since the right-hand side is a variable, this would not be allowed in Arts and Giesl's approach [10].

Collapsing pairs are essential in our approach. They express that infinite context-sensitive rewrite sequences can involve not only the kind of recursion that is represented by the *usual* dependency pairs but also a new kind of recursion that is *hidden* inside the nonreplacing (or *frozen*) parts of the terms involved in the infinite sequence. The *activation* of such *delayed* recursions is due to the presence of *migrating* variables within a rule  $l \rightarrow r$  which is used in the sequence. Migrating variables are those that are not replacing in the left-hand side  $l$  but that *become* replacing in the right-hand side  $r$ .

**Example 4** (Continuing Example 2). The following *collapsing* pairs are *context-sensitive dependency pairs* for the CS-TRS in Example 1:

$$\text{TAIL}(\text{cons}(x, xs)) \rightarrow xs \tag{25}$$

$$\text{TAKE}(s(n), \text{cons}(x, xs)) \rightarrow xs \tag{26}$$

Note that variable  $xs$  is  $\mu$ -replacing in the right-hand sides of the rules  $\text{tail}(\text{cons}(x, xs)) \rightarrow xs$  and  $\text{take}(s(n), \text{cons}(x, xs)) \rightarrow \text{consF}(x, \text{take}(n, xs))$  but it is *non- $\mu$ -replacing* in the corresponding left-hand sides.

### 1.3. Plan of the paper

We have argued that termination of CSR is an interesting and challenging topic of research with a good number of practical applications. The results, techniques, and tools that derive from our work can be of interest to a sufficiently wide audience. The material in this paper will be more familiar, however, to those specialists who are interested in termination (in general) and in *how* to prove termination of CSR in particular. Throughout the paper, however, we made a serious effort to provide sufficient intuition and informal descriptions for our main definitions and results.

After Section 2, the paper is structured in three main parts:

1. Section 3 provides appropriate notions of *minimal* non- $\mu$ -terminating terms and introduces the main properties of such terms. We introduce the notion of *hidden term* and investigate the structure of infinite context-sensitive rewrite sequences starting from minimal non- $\mu$ -terminating terms. This analysis is essential in order to provide an appropriate definition of context-sensitive dependency pair and the related notions of chains, graphs, etc.
2. We define the notions of *context-sensitive dependency pair* and *context-sensitive chain of pairs* and show how to use them to *characterize* termination of CSR. Sections 4 and 5 introduce the general framework to compute and use context-sensitive dependency pairs to prove termination of CSR. The introduction of dependency pairs of a new kind (the *collapsing* dependency pairs, as in Example 3) leads to a notion of context-sensitive dependency *chain*, which is quite different from the standard one. In Section 6, we prove that our *context-sensitive dependency pair approach* fully characterizes termination of CSR.
3. We describe a suitable *framework* for dealing with proofs of termination of CSR by using these results. Section 7 adapts the *dependency pair framework* [33,35] to CSR by defining appropriate notions of *CS problem* and *CS processor* that rely on the results obtained in the second part of the paper. Section 8 introduces the notion of *context-sensitive (dependency) graph* and the associated CS processor. Section 9 describes CS processors for removing or transforming collapsing pairs. Section 10 investigates the use of term orderings in processors. Section 11 adapts Hirokawa and Middeldorp's *subterm criterion* [38]. Section 12 adapts *narrowing transformation* of pairs in [35].

Experiments are reported in Section 13. Sections 14 and 15 discuss related work. Section 16 concludes.

## 2. Preliminaries

This section collects a number of definitions and notations about term rewriting. More details and missing notions can be found in [11,58,61].

Let  $A$  be a set and  $R \subseteq A \times A$  be a binary relation on  $A$ . We denote the transitive closure of  $R$  by  $R^+$  and its reflexive and transitive closure by  $R^*$ . We say that  $R$  is *terminating* (*strongly normalizing*) if there is no infinite sequence  $a_1 R a_2 R a_3 \dots$ . A reflexive and transitive relation  $R$  is a quasi-ordering.

Given relations  $R$  and  $R'$  over the same set  $A$ , we define its *composition*  $R \circ R'$  as follows: for all  $a, b \in A$ ,  $a (R \circ R') b$  if there is  $c \in A$  such that  $a R c$  and  $c R' b$ .

### 2.1. Signatures, terms, and positions

Throughout the paper,  $\mathcal{X}$  denotes a countable set of variables and  $\mathcal{F}$  denotes a signature, i.e., a set of function symbols  $\{\varepsilon, g, \dots\}$ , each having a fixed arity given by a mapping  $ar : \mathcal{F} \rightarrow \mathbb{N}$ . The set of terms built from  $\mathcal{F}$  and  $\mathcal{X}$  is  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ .  $\text{Var}(t)$  is the set of variables occurring in a term  $t$ . A term  $t$  is *ground* if it contains no variable (i.e.,  $\text{Var}(t) = \emptyset$ ). A term is said to be *linear* if it has no multiple occurrences of a single variable.

Terms are viewed as labelled trees in the usual way. Positions  $p, q, \dots$  are represented by chains of positive natural numbers used to address subterms of  $t$ . We denote the empty chain by  $\Lambda$ . Given positions  $p, q$ , we denote their concatenation as  $p \cdot q$ . Positions are ordered by the standard prefix ordering:  $p \leq q$  if  $\exists q'$  such that  $q = p \cdot q'$ . If  $p$  is a position, and  $Q$  is a set of positions, then  $p \cdot Q = \{p \cdot q \mid q \in Q\}$ . The set of positions of a term  $t$  is  $\text{Pos}(t)$ . Positions of nonvariable symbols in  $t$  are denoted as  $\text{Pos}_{\mathcal{F}}(t)$ , and  $\text{Pos}_{\mathcal{X}}(t)$  are the positions of variables. The subterm at position  $p$  of  $t$  is denoted as  $t|_p$ , and  $t[s]_p$  is the term  $t$  with the subterm at position  $p$  replaced by  $s$ .

We write  $s \triangleright t$ , read  $t$  is a *subterm* of  $s$ , if  $t = s|_p$  for some  $p \in \text{Pos}(s)$  and  $s \triangleright t$  if  $s \triangleright t$  and  $s \neq t$ . We write  $s \not\triangleright t$  and  $s \not\triangleright t$  for the negation of the corresponding properties. The symbol labeling the root of  $t$  is denoted as  $\text{root}(t)$ . A *context* is a term  $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{X})$  with a 'hole'  $\square$  (a fresh constant symbol). We write  $C[\ ]_p$  to denote that there is a (usually single) hole  $\square$  at position  $p$  of  $C$ . Generally, we write  $C[\ ]$  to denote an arbitrary context and make the position of the hole explicit only if necessary.  $C[\ ] = \square$  is called the *empty context*.

### 2.2. Substitutions, renamings, and unifiers

A substitution is a mapping  $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$ . Denote as  $\varepsilon$  the 'identity' substitution:  $\varepsilon(x) = x$  for all  $x \in \mathcal{X}$ . The set  $\text{Dom}(\sigma) = \{x \in \mathcal{X} \mid \sigma(x) \neq x\}$  is called the *domain* of  $\sigma$ .

**Remark 1.** We do *not* impose that the domain of the substitutions be finite. This is usual practice in the dependency pair approach, where a single substitution is used to instantiate an infinite number of variables coming from renamed versions of the dependency pairs (see below).

A *renaming* is an injective substitution  $\rho$  such that  $\rho(x) \in \mathcal{X}$  for all  $x \in \mathcal{X}$ . A substitution  $\sigma$  such that  $\sigma(s) = \sigma(t)$  for two terms  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is called a *unifier* of  $s$  and  $t$ ; we also say that  $s$  and  $t$  unify (with substitution  $\sigma$ ). If two terms  $s$  and  $t$  unify, then there is a unique *most general unifier*  $\sigma$  (up to renaming of variables) such that for every other unifier  $\tau$ , there is a substitution  $\theta$  such that  $\theta \circ \sigma = \tau$ .

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

### 2.3. Rewrite systems and term rewriting

A rewrite rule is an ordered pair  $(l, r)$ , written  $l \rightarrow r$ , with  $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $l \notin \mathcal{X}$  and  $\text{Var}(r) \subseteq \text{Var}(l)$ . The left-hand side (*lhs*) of the rule is  $l$ , and the right-hand side (*rhs*) is  $r$ . A rewrite rule  $l \rightarrow r$  is said to be *collapsing* if  $r \in \mathcal{X}$ . A *Term Rewriting System* (TRS) is a pair  $\mathcal{R} = (\mathcal{F}, R)$ , where  $R$  is a set of rewrite rules. We often use  $\emptyset$  to denote TRSs whose set of rules  $R$  is empty. Given TRSs  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{R}' = (\mathcal{F}', R')$ , we let  $\mathcal{R} \cup \mathcal{R}'$  be the TRS  $(\mathcal{F} \cup \mathcal{F}', R \cup R')$ . An instance  $\sigma(l)$  of a *lhs*  $l$  of a rule is called a *redex*. Given  $\mathcal{R} = (\mathcal{F}, R)$ , we consider  $\mathcal{F}$  as the disjoint union  $\mathcal{F} = \mathcal{C} \uplus \mathcal{D}$  of symbols  $c \in \mathcal{C}$  (called *constructors*) and symbols  $f \in \mathcal{D}$  (called *defined functions*), where  $\mathcal{D} = \{\text{root}(l) \mid l \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} - \mathcal{D}$ .

**Example 5.** Consider again the TRS in Example 1. The symbols `evenNs`, `oddNs`, `incr`, `take`, `zip`, `tail`, `rep2`, `add`, `prod`, `prodFrac`, `prodOfFrac`s, and `halfPi` are defined. Symbols `s`, `0`, `cons`, `consF`, `nil`, and `frac` are constructors.

We often write  $l \rightarrow r \in \mathcal{R}$  instead of  $l \rightarrow r \in R$  to express that the rule  $l \rightarrow r$  is a rule of  $\mathcal{R}$ . A term  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  rewrites to  $t$  (at position  $p$ ), written  $s \xrightarrow{p} t$  (or just  $s \rightarrow t$ , or  $s \rightarrow t$ ), if  $s|_p = \sigma(l)$  and  $t = s[\sigma(r)]_p$ , for some rule  $l \rightarrow r \in \mathcal{R}$ ,  $p \in \text{Pos}(s)$  and substitution  $\sigma$ . We write  $s \xrightarrow{p} t$  if  $s \xrightarrow{q} t$  for some  $q > p$ . A TRS  $\mathcal{R}$  is *terminating* if its one step rewrite relation  $\rightarrow_{\mathcal{R}}$  is terminating.

### 2.4. Context-sensitive rewriting

A mapping  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  is a *replacement map* (or  $\mathcal{F}$ -map) if for all symbols  $f \in \mathcal{F}$ ,  $\mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$  [44]. Let  $M_{\mathcal{F}}$  be the set of all  $\mathcal{F}$ -maps (or  $M_{\mathcal{R}}$  for the  $\mathcal{F}$ -maps of a TRS  $(\mathcal{F}, R)$ ). Let  $\mu_{\top}$  be the replacement map given by  $\mu_{\top}(f) = \{1, \dots, \text{ar}(f)\}$  for all  $f \in \mathcal{F}$  (i.e., no replacement restrictions are specified).

A binary relation  $R$  on terms is  $\mu$ -monotonic if, for all  $f \in \mathcal{F}$ ,  $i \in \mu(f)$ , and  $s, t, t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $f(t_1, \dots, t_{i-1}, s, t_{i+1}, \dots, t_k) R f(t_1, \dots, t_{i-1}, t, t_{i+1}, \dots, t_k)$  whenever  $s R t$ . If  $R$  is  $\mu_{\top}$ -monotonic, we just say that  $R$  is *monotonic*.

The set of  $\mu$ -replacing positions  $\text{Pos}^{\mu}(t)$  of  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is:  $\text{Pos}^{\mu}(t) = \{\Lambda\}$  if  $t \in \mathcal{X}$ , and  $\text{Pos}^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(\text{root}(t))} i.\text{Pos}^{\mu}(t_i)$  if  $t \notin \mathcal{X}$ . Note that  $\text{Pos}^{\mu}(t)$  (as  $\text{Pos}(t)$ ) is *prefix closed*. When no replacement map is made explicit, the  $\mu$ -replacing positions are often called *active*; and the non- $\mu$ -replacing ones are often called *frozen*. The following results about CSR are often used without any explicit mention.

**Proposition 1** [44]. *Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $p = q \cdot q' \in \text{Pos}(t)$ . Then  $p \in \text{Pos}^{\mu}(t)$  iff  $q \in \text{Pos}^{\mu}(t) \wedge q' \in \text{Pos}^{\mu}(t|_q)$ .*

The chain of symbols lying on positions above/on  $p \in \text{Pos}(t)$  is  $\text{prefix}_t(\Lambda) = \text{root}(t)$ ,  $\text{prefix}_t(i \cdot p) = \text{root}(t).\text{prefix}_{t_i}(p)$ . The strict prefix *spre*fix is  $\text{spre}fix_t(\Lambda) = \Lambda$ ,  $\text{spre}fix_t(p \cdot i) = \text{prefix}_t(p)$ , i.e., the last symbol in  $\text{prefix}_t(p \cdot i)$  is *removed*. Although  $\text{spre}fix_t(p)$  is a sequence, when the ordering of symbols in  $\text{prefix}_t(p)$  does not matter, we often use the standard set-theoretic notation (e.g., inclusion as in  $\text{spre}fix_t(p) \subseteq \mathcal{F}$ ) with the obvious meaning.

**Proposition 2** [44]. *If  $p \in \text{Pos}(t) \cap \text{Pos}(s)$  and  $\text{spre}fix_t(p) = \text{spre}fix_s(p)$ , then  $p \in \text{Pos}^{\mu}(t) \Leftrightarrow p \in \text{Pos}^{\mu}(s)$ .*

The  $\mu$ -replacing subterm relation  $\triangleright_{\mu}$  is given by  $s \triangleright_{\mu} t$  if there is  $p \in \text{Pos}^{\mu}(s)$  such that  $t = s|_p$ . We write  $s \triangleright_{\mu} t$  if  $s \triangleright_{\mu} t$  and  $s \neq t$ . We write  $s \triangleright_{\mu} t$  to denote that  $t$  is a non- $\mu$ -replacing (hence strict) subterm of  $s$ :  $s \triangleright_{\mu} t$  if there is  $p \in \text{Pos}(s) - \text{Pos}^{\mu}(s)$  such that  $t = s|_p$ . The set of  $\mu$ -replacing variables of a term  $t$ , i.e., variables occurring at some  $\mu$ -replacing position in  $t$ , is  $\text{Var}^{\mu}(t) = \{x \in \text{Var}(t) \mid t \triangleright_{\mu} x\}$ . The set of non- $\mu$ -replacing variables of  $t$ , i.e., variables occurring at some non- $\mu$ -replacing position in  $t$ , is  $\text{Var}^{\neq \mu}(t) = \{x \in \text{Var}(t) \mid t \not\triangleright_{\mu} x\}$ . Note that  $\text{Var}^{\mu}(t)$  and  $\text{Var}^{\neq \mu}(t)$  do not need to be disjoint (when  $t$  is not linear).

A pair  $(\mathcal{R}, \mu)$  where  $\mathcal{R}$  is a TRS and  $\mu \in M_{\mathcal{R}}$  is often called a CS-TRS. In *context-sensitive rewriting*, we (only) contract  $\mu$ -replacing redexes:  $s$   $\mu$ -rewrites to  $t$ , written  $s \xrightarrow{p}_{\mathcal{R}, \mu} t$  (or  $s \xrightarrow{\mu} t$ ,  $s \xrightarrow{\mu} t$  and  $\text{even } s \xrightarrow{\mu} t$ ), if  $s \xrightarrow{p}_{\mathcal{R}} t$  and  $p \in \text{Pos}^{\mu}(s)$ .

**Example 6.** Consider  $\mathcal{R}$  and  $\mu$  as in Example 1. Then, we have:

$$\text{evenNs} \xrightarrow{\mu} \text{cons}(0, \text{incr}(\text{oddNs})) \not\xrightarrow{\mu} \text{cons}(0, \text{incr}(\text{incr}(\text{evenNs})))$$

Since the second argument of  $\text{cons}$  is not  $\mu$ -replacing, we have  $2 \notin \text{Pos}^{\mu}(\text{cons}(0, \text{incr}(\text{oddNs})))$ . Thus, redex  $\text{oddNs}$  cannot be  $\mu$ -rewritten.

A term  $t$  is  $\mu$ -terminating (or  $(\mathcal{R}, \mu)$ -terminating, if we want an explicit reference to the involved TRS  $\mathcal{R}$ ) if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \xrightarrow{\mu} t_2 \xrightarrow{\mu} t_3 \cdots \xrightarrow{\mu} t_n \xrightarrow{\mu} t_{n+1} \cdots$  starting from  $t$ . A TRS  $\mathcal{R}$  is  $\mu$ -terminating if  $\xrightarrow{\mu}$  is terminating.

A term  $s$   $\mu$ -narrows to a term  $t$  (written  $s \xrightarrow{\mu, \theta} t$ ), if there is a nonvariable  $\mu$ -replacing position  $p \in \text{Pos}_{\mathcal{F}}^{\mu}(s)$  and a rule  $l \rightarrow r$  in  $\mathcal{R}$  (sharing no variable with  $s$ ) such that  $s|_p$  and  $l$  unify with the most general unifier  $\theta$  and  $t = \theta(s[r]_p)$ . The following definition is used in Section 10.2.

**Definition 1** [26]. Let  $\mathcal{F}$  be a signature and  $\mu \in M_{\mathcal{F}}$ . The  $\mu$ -replacing projection TRS  $\text{Emb}^{\mu}(\mathcal{F})$  consists of the following rules:

$$\{f(x_1, \dots, x_k) \rightarrow x_i \mid f \in \mathcal{F}, i \in \mu(f)\}$$

### 3. Minimal non- $\mu$ -terminating terms and infinite $\mu$ -rewrite sequences

Given a TRS  $\mathcal{R} = (\mathcal{C} \cup \mathcal{D}, R)$ , the *minimal* nonterminating terms associated to  $\mathcal{R}$  are nonterminating terms  $t$  whose proper subterms  $u$  (i.e.,  $t \triangleright u$ ) are terminating;  $\mathcal{T}_{\infty}$  is the set of minimal nonterminating terms associated to  $\mathcal{R}$  [38,40]. Minimal nonterminating terms have two important properties:

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

As discussed in [38], considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term  $t \in \mathcal{T}_{\infty}$  can be helpful to come to the notion of dependency pair [10]. Such sequences proceed as follows:

**Proposition 3** [38, Lemma 1]. Let  $\mathcal{R} = (\mathcal{C} \cup \mathcal{D}, R)$  be a TRS. For all  $t \in \mathcal{T}_{\infty}$ , there exist  $l \rightarrow r \in R$ , a substitution  $\sigma$  and a term  $u \in \mathcal{T}_{\infty}$  such that  $\text{root}(u) \in \mathcal{D}$ ,  $t \xrightarrow{\geq \Delta} \sigma(l) \xrightarrow{\Delta} \sigma(r) \triangleright u$ , and there is a nonvariable subterm  $v$  of  $r$ ,  $r \triangleright v$ , such that  $u = \sigma(v)$ .

In the following, we show how to generalize these notions and results to CSR.

#### 3.1. Minimal non- $\mu$ -terminating terms

Before starting our discussion about (minimal) non- $\mu$ -terminating terms, we provide an obvious auxiliary result about  $\mu$ -terminating terms.<sup>2</sup>

**Lemma 1.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is  $\mu$ -terminating, then:

1. If  $s \triangleright_{\mu} t$ , then  $t$  is  $\mu$ -terminating.
2. If  $s \xrightarrow{*}_{\mathcal{R}, \mu} t$ , then  $t$  is  $\mu$ -terminating.

Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$  and a replacement map  $\mu \in M_{\mathcal{F}}$ , maybe the simplest extension to CSR of the notion of minimal term for unrestricted rewriting (i.e.,  $\mathcal{T}_{\infty}$ ), is the following: let  $\mathcal{T}_{\infty, \mu}$  be a set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{T}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict subterm  $u$  (i.e.,  $t \triangleright u$ ) is  $\mu$ -terminating. It is obvious that  $\text{root}(t) \in \mathcal{D}$  for all  $t \in \mathcal{T}_{\infty, \mu}$ . We also have the following:

**Lemma 2.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is not  $\mu$ -terminating, then there is a subterm  $t$  of  $s$  ( $s \triangleright t$ ) such that  $t \in \mathcal{T}_{\infty, \mu}$ .

<sup>2</sup> For the sake of readability, the missing proofs of the technical results in this section have been moved to Appendix A.

Unfortunately, there can be non- $\mu$ -terminating terms having no  $\mu$ -replacing subterm in  $\mathcal{T}_{\infty, \mu}$ .

**Example 7.** Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 3 and  $s = \varepsilon(c(\varepsilon(a)))$ . Note that  $s$  is not  $\mu$ -terminating, but  $s \notin \mathcal{T}_{\infty, \mu}$  because  $\varepsilon(c(\varepsilon(a))) \triangleright_{\mu} \varepsilon(a)$  and  $\varepsilon(a)$  is not  $\mu$ -terminating. Note that  $\varepsilon(c(\varepsilon(a))) \triangleright_{\mu} \varepsilon(a)$ . The only  $\mu$ -replacing strict subterm of  $s$  is  $c(\varepsilon(a))$ , which is  $\mu$ -terminating, i.e.,  $c(\varepsilon(a)) \notin \mathcal{T}_{\infty, \mu}$ .

Therefore, minimal non- $\mu$ -terminating terms are not the most natural ones because they could occur at non- $\mu$ -replacing positions, where no  $\mu$ -rewriting step is possible. Thus, this simple notion would not lead to an appropriate generalization of Proposition 3 to CSR. There is a suitable generalization of Proposition 3 to CSR (see Proposition 5) based on the following notion.

**Definition 2 (Minimal non- $\mu$ -terminating term).** Let  $\mathcal{M}_{\infty, \mu}$  be a set of minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty, \mu}$  if  $t$  is non- $\mu$ -terminating and every strict  $\mu$ -replacing subterm  $t'$  of  $t$  (i.e.,  $t \triangleright_{\mu} t'$ ) is  $\mu$ -terminating.

Note that  $\mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ . In the following, we often say that terms in  $\mathcal{T}_{\infty, \mu}$  are *strongly minimal* non- $\mu$ -terminating; we use them in Section 3.4. Now, we have the following:

**Lemma 3.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is not  $\mu$ -terminating, then there is a  $\mu$ -replacing subterm  $t$  of  $s$  such that  $t \in \mathcal{M}_{\infty, \mu}$ .

Obviously, if  $t \in \mathcal{M}_{\infty, \mu}$ , then  $\text{root}(t)$  is a defined symbol. Since  $\mu$ -terminating terms are preserved under  $\mu$ -rewriting (Lemma 1), it follows that  $\mathcal{M}_{\infty, \mu}$  is preserved under *inner*  $\mu$ -rewritings in the following sense.

**Lemma 4.** Let  $\mathcal{R}$  be a TRS,  $\mu \in M_{\mathcal{R}}$ , and  $t \in \mathcal{M}_{\infty, \mu}$ . If  $t \xrightarrow{>\Lambda}^* u$  and  $u$  is non- $\mu$ -terminating, then  $u \in \mathcal{M}_{\infty, \mu}$ .

Lemma 4 does not hold for  $\mathcal{T}_{\infty, \mu}$ : consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 3. Note that  $\varepsilon(a) \in \mathcal{T}_{\infty, \mu}$  and  $\varepsilon(a) \xrightarrow{>\Lambda} \varepsilon(c(\varepsilon(a)))$ . Although  $\varepsilon(c(\varepsilon(a)))$  is not  $\mu$ -terminating,  $\varepsilon(c(\varepsilon(a))) \notin \mathcal{T}_{\infty, \mu}$ , as shown in Example 7.

### 3.2. Hidden terms in minimal $\mu$ -rewrite sequences

Given a CS-TRS  $(\mathcal{R}, \mu)$ , the *hidden terms* are nonvariable terms occurring on some frozen position in the right-hand side of some rule of  $\mathcal{R}$ . As we show in the next section, they play an important role in infinite minimal  $\mu$ -rewrite sequences associated to  $\mathcal{R}$ .

**Definition 3 (Hidden symbols and terms).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . We say that  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$  is a *hidden term* if there is a rule  $l \rightarrow r \in R$  such that  $r \triangleright_{\mu} t$ . Let  $\mathcal{HT}(\mathcal{R}, \mu)$  (or just  $\mathcal{HT}$ , if no confusion arises) be the set of all hidden terms in  $(\mathcal{R}, \mu)$ . We say that  $f \in \mathcal{F}$  is a *hidden symbol* if it occurs in a hidden term. Let  $\mathcal{H}(\mathcal{R}, \mu)$  (or just  $\mathcal{H}$ ) be the set of all hidden symbols in  $(\mathcal{R}, \mu)$ .

In the following, we also use  $\mathcal{DHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D}\}$  for the set of hidden terms which are rooted by a *defined* symbol.

**Example 8.** For  $\mathcal{R}$  and  $\mu$  as in Example 1, the maximal hidden terms are  $\text{incr}(\text{oddNs})$ ,  $\text{incr}(x)$ ,  $\text{zip}(xs, ys)$ , and  $\text{cons}(x, \text{rep2}(xs))$ . The hidden symbols are  $\text{incr}$ ,  $\text{oddNs}$ ,  $\text{zip}$ ,  $\text{cons}$ , and  $\text{rep2}$ . Finally,  $\mathcal{DHT}(\mathcal{R}, \mu) = \{\text{oddNs}, \text{incr}(\text{oddNs}), \text{incr}(x), \text{zip}(xs, ys), \text{rep2}(xs)\}$ .

The following lemma says that frozen subterms  $t$  in the contractum  $\sigma(r)$  of a redex  $\sigma(l)$  that do not contain  $t$  are (at least partly) ‘introduced’ by a hidden term in the right-hand side  $r$  of the involved rule  $l \rightarrow r$ .

**Lemma 5.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $\sigma$  be a substitution. If there is a rule  $l \rightarrow r \in R$  such that  $\sigma(l) \not\triangleright_{\mu} t$  and  $\sigma(r) \triangleright_{\mu} t$ , then there is no  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright_{\mu} t$ . Furthermore, there is a term  $t' \in \mathcal{HT}$  such that  $r \triangleright_{\mu} t'$  and  $\sigma(t') = t$ .

The following lemma establishes that minimal non- $\mu$ -terminating and non- $\mu$ -replacing subterms that occur in a  $\mu$ -rewrite sequence involving only minimal terms come directly from the first term in the sequence or are instances of a hidden term.

**Lemma 6.** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $A$  be a  $\mu$ -rewrite sequence  $t_1 \hookrightarrow t_2 \hookrightarrow \dots \hookrightarrow t_n$  with  $t_i \in \mathcal{M}_{\infty, \mu}$  for all  $i$ ,  $1 \leq i \leq n$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_1 \not\triangleright_{\mu} t$  and  $t_n \triangleright_{\mu} t$ , then  $t = \sigma(s)$  for some  $s \in \mathcal{DHT}$  and substitution  $\sigma$ .

We use the previous results to investigate infinite sequences that combine  $\mu$ -rewriting steps on minimal non- $\mu$ -terminating terms and the extraction of such subterms as  $\mu$ -replacing subterms of (instances of) right-hand sides of the rules.

**Proposition 4.** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Consider a finite or infinite sequence of the form  $t_1 \xrightarrow{\Delta} s_1 \triangleright_{\mu} t'_2 \xrightarrow{\Delta} t_2 \xrightarrow{\Delta} s_2 \triangleright_{\mu} t'_3 \xrightarrow{\Delta} t_3 \dots$  with  $t_i, t'_i \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_i \triangleright_{\mu} t$  for some  $i \geq 1$ , then  $t_1 \triangleright_{\mu} t$  or  $t = \sigma(s)$  for some  $s \in \mathcal{DHT}$  and substitution  $\sigma$ .

### 3.3. Infinite $\mu$ -rewrite sequences starting from minimal terms

The following proposition establishes that, given a minimal non- $\mu$ -terminating term  $t \in \mathcal{M}_{\infty, \mu}$ , there are only two ways for an infinite  $\mu$ -rewrite sequence to proceed. The first one is by using 'visible' parts of the rules that correspond to  $\mu$ -replacing nonvariable subterms in the right-hand sides that are rooted by a defined symbol. The second one is by showing up 'hidden' non- $\mu$ -terminating subterms that are activated by migrating variables in a rule  $l \rightarrow r$ , i.e., variables  $x \in \text{Var}^{\mu}(r) - \text{Var}^{\mu}(l)$  that are not  $\mu$ -replacing in the left-hand side  $l$  but become  $\mu$ -replacing in the right-hand side  $r$ .

**Proposition 5.** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then, for all  $t \in \mathcal{M}_{\infty, \mu}$ , there exist  $l \rightarrow r \in \mathcal{R}$ , a substitution  $\sigma$ , and a term  $u \in \mathcal{M}_{\infty, \mu}$  such that  $t \xrightarrow{\Delta} \sigma(l) \xrightarrow{\Delta} \sigma(r) \triangleright_{\mu} u$  and either

1. there is a nonvariable  $\mu$ -replacing subterm  $s$  of  $r$ ,  $r \triangleright_{\mu} s$ , such that  $u = \sigma(s)$ , or
2. there is  $x \in \text{Var}^{\mu}(r) - \text{Var}^{\mu}(l)$  such that  $\sigma(x) \triangleright_{\mu} u$ .

**Proof.** Consider an infinite  $\mu$ -rewrite sequence starting from  $t$ . By definition of  $\mathcal{M}_{\infty, \mu}$ , all proper  $\mu$ -replacing subterms of  $t$  are  $\mu$ -terminating. Therefore,  $t$  has an inner reduction to an instance  $\sigma(l)$  of the left-hand side of a rule  $l \rightarrow r$  of  $\mathcal{R}$ :  $t \xrightarrow{\Delta} \sigma(l) \xrightarrow{\Delta} \sigma(r)$  and  $\sigma(r)$  is not  $\mu$ -terminating. Thus, we can write  $t = f(t_1, \dots, t_k)$  and  $\sigma(l) = f(l_1, \dots, l_k)$  for some  $k$ -ary defined symbol  $f$ , and  $t_i \hookrightarrow \sigma(l_i)$  for all  $i$ ,  $1 \leq i \leq k$ . Since all  $t_i$  are  $\mu$ -terminating for  $i \in \mu(f)$ , by Lemma 1,  $\sigma(l_i)$  and all its  $\mu$ -replacing subterms are also  $\mu$ -terminating. In particular,  $\sigma(y)$  is  $\mu$ -terminating for all  $\mu$ -replacing variables  $y$  in  $l$ :  $y \in \text{Var}^{\mu}(l)$ . Since  $\sigma(r)$  is non- $\mu$ -terminating, by Lemma 3, it contains a  $\mu$ -replacing subterm  $u \in \mathcal{M}_{\infty, \mu}$ :  $\sigma(r) \triangleright_{\mu} u$ , i.e., there is a position  $p \in \text{Pos}^{\mu}(\sigma(r))$  such that  $\sigma(r)|_p = u$ . We consider two cases:

1. If  $p \in \text{Pos}_{\mathcal{F}}(r)$  is a nonvariable position of  $r$ , then there is a  $\mu$ -replacing nonvariable subterm  $s$  of  $r$  (i.e.,  $p \in \text{Pos}_{\mathcal{F}}^{\mu}(r)$  and  $s = r|_p \notin \mathcal{X}$ ), such that  $u = \sigma(s)$ .
2. If  $p \notin \text{Pos}_{\mathcal{F}}(r)$ , then there is a  $\mu$ -replacing variable position  $q \in \text{Pos}^{\mu}(r) \cap \text{Pos}_{\mathcal{X}}(r)$  such that  $q \leq p$ . Let  $x \in \text{Var}^{\mu}(r)$  be such that  $r|_q = x$ . Then,  $\sigma(x) \triangleright_{\mu} u$ , and  $\sigma(x)$  is not  $\mu$ -terminating: since  $u \in \mathcal{M}_{\infty, \mu}$  is not  $\mu$ -terminating, by Lemma 1,  $\sigma(x)$  is not  $\mu$ -terminating. Since  $\sigma(y)$  is  $\mu$ -terminating for all  $y \in \text{Var}^{\mu}(l)$ , we conclude that  $x \in \text{Var}^{\mu}(r) - \text{Var}^{\mu}(l)$ .  $\square$

Proposition 5 entails the following result, which establishes some properties of infinite sequences starting from minimal non- $\mu$ -terminating terms.

**Corollary 1.** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . For all  $t \in \mathcal{M}_{\infty, \mu}$ , there is an infinite sequence

$$t \xrightarrow{\Delta} \sigma_1(l_1) \xrightarrow{\Delta} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\Delta} \sigma_2(l_2) \xrightarrow{\Delta} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{\Delta} \dots$$

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

1.  $t_i = \sigma_i(s_i)$  for some nonvariable subterm  $s_i$  such that  $r_i \triangleright_{\mu} s_i$ , or
2.  $\sigma_i(x_i) \triangleright_{\mu} t_i$  for some  $x_i \in \text{Var}^{\mu}(r_i) - \text{Var}^{\mu}(l_i)$ .

**Remark 2.** The  $(\hookrightarrow_{\mu} \cup \triangleright_{\mu})$ -sequence in Corollary 1 can be easily viewed as an infinite  $\mu$ -rewrite sequence by just introducing appropriate contexts  $C_i[\ ]_{p_i}$  with  $\mu$ -replacing holes: since  $\sigma_i(r_i) \triangleright_{\mu} t_i$ , there is  $p_i \in \text{Pos}^{\mu}(\sigma_i(r_i))$  such that  $\sigma_i(r_i) = \sigma_i(r_i)[t_i]_{p_i}$ ; just take  $C_i[\ ]_{p_i} = \sigma_i(r_i)[\ ]_{p_i}$ . Hence:

$$t \hookrightarrow^* \sigma_1(l_1) \hookrightarrow C_1[t_1]_{p_1} \hookrightarrow^* C_1[\sigma_2(l_2)]_{p_1} \hookrightarrow C_1[C_2[t_2]_{p_2}]_{p_1} \hookrightarrow^* \dots$$

Note that, e.g.,  $p_1 \cdot p_2 \in \text{Pos}^\mu(C_1[C_2[t_2]_{p_2}]_{p_1})$  (use Proposition 1).

### 3.4. Infinite $\mu$ -rewrite sequences starting from strongly minimal terms

In the following, we consider a function  $\text{REN}^\mu$  which *independently* renames all occurrences of  $\mu$ -replacing variables within a term  $t$  by using new fresh variables that are not in  $\text{Var}(t)$ :

- $\text{REN}^\mu(x) = y$  if  $x$  is a variable, where  $y$  is intended to be a fresh new variable that has not yet been used;
- $\text{REN}^\mu(f(t_1, \dots, t_k)) = f([t_1]_1^f, \dots, [t_k]_k^f)$  for every  $k$ -ary symbol  $f$ , where given a term  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $[s]_i^f = \text{REN}^\mu(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ .

Note that  $\text{REN}^\mu(t)$  keeps variables at non- $\mu$ -replacing positions untouched. Note also that  $\text{REN}^\mu$  is *not* a substitution: it replaces the  $n(x)$  different  $\mu$ -replacing occurrences of the same variable  $x$  by *different* variables  $x_1, \dots, x_{n(x)}$ . Clearly,  $t = \theta(\text{REN}^\mu(t))$  for some substitution  $\theta$  which just identifies the variables introduced by  $\text{REN}^\mu$  (i.e.,  $\theta(x_i) = x$  for all  $1 \leq i \leq n(x)$ ). The use of  $\text{REN}^\mu$  together with  $\mu$ -narrowability yields a necessary condition for reducibility of terms under some instantiations which is used in our development.

**Proposition 6.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$  be a nonvariable term and  $\sigma$  be a substitution. If  $\sigma(t) \xrightarrow{>\Delta}^* \sigma(l)$  for some (possibly renamed) rule  $l \rightarrow r \in \mathcal{R}$ , then  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable.

**Proof.** We can write the sequence from  $\sigma(t)$  to  $\sigma(l)$  as follows:  $\sigma(t) = t_1 \xrightarrow{>\Delta} t_2 \xrightarrow{>\Delta} \dots \xrightarrow{>\Delta} t_m = \sigma(l)$  for some  $m \geq 1$ . We proceed by induction on  $m$ .

1. If  $m = 1$ , then  $\sigma(t) = \sigma(l)$ . Since  $t \notin \mathcal{X}$ ,  $t$  is  $\mu$ -narrowable (at the root position) using the rule  $l \rightarrow r$ . Since  $t = \theta(\text{REN}^\mu(t))$  for some substitution  $\theta$ , we have  $\sigma(t) = \sigma(\theta(\text{REN}^\mu(t))) = \sigma(l)$ . Since we can assume that the new variables instantiated by  $\theta$  are not in  $l$ , we have  $\sigma(\theta(l)) = \sigma(l)$ . Thus,  $\text{REN}^\mu(t)$  and  $l$  unify with mgu  $\sigma \circ \theta$ . Since  $t \notin \mathcal{X}$ , implies that  $\text{REN}^\mu(t) \notin \mathcal{X}$ ,  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable at the root position using the same rule  $l \rightarrow r$ .
2. If  $m > 1$ , then we have  $t_1 \xrightarrow{>\Delta} t_2 \xrightarrow{>\Delta}^* \sigma(l)$ . We consider two cases according to the position  $p \in \text{Pos}^\mu(t_1)$  where the  $\mu$ -rewrite step  $t_1 \xrightarrow{>\Delta} t_2$  is performed (note that  $t_1 = \sigma(t)$  by assumption).
  - (a) If  $p \in \text{Pos}_{\mathcal{F}}^\mu(t)$ , then there is a rule  $l' \rightarrow r'$  and a substitution  $\theta$  such that  $\sigma(t)|_p = \sigma(t|_p) = \theta(l')$ . Again, we have  $\sigma(t|_p) = \sigma(l')$ , i.e.,  $t$  is  $\mu$ -narrowable at position  $p$  using rule  $l' \rightarrow r'$  and (reasoning as above), we conclude that  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable.
  - (b) If  $p \notin \text{Pos}_{\mathcal{F}}^\mu(t)$ , then there is a  $\mu$ -replacing variable position  $q \in \text{Pos}_{\mathcal{X}}^\mu(t)$  of  $t$  such that  $t|_q = x \in \text{Var}^\mu(t)$ ,  $q \leq p$  and  $\sigma(x) \hookrightarrow_{\mu} t_2|_q$ . Therefore,  $t_1 = \sigma(t|_x]_q) = \sigma(t)[\sigma(x)]_q$  and  $t_2 = \sigma(t)[t_2|_q]_q = \sigma'(t')$  for a term  $t' = t|_y]_q$  where  $y$  is a new fresh variable  $y \notin \text{Var}(t)$  and a substitution  $\sigma'$  given by  $\sigma'(y) = t_2|_q$  and  $\sigma'(z) = \sigma(z)$  for all  $z \in \text{Var}(t)$  (including  $x$ ). Clearly,

$$\sigma'(t') = \sigma'(t|_y]_q) = \sigma'(t)[\sigma'(y)]_q = \sigma(t)[t_2|_q]_q = t_2.$$

By the induction hypothesis,  $\text{REN}^\mu(t')$  is  $\mu$ -narrowable. Since  $t$  and  $t'$  only differ in a single variable, we can assume that  $\text{REN}^\mu(t') = \text{REN}^\mu(t)$ . Thus, we conclude that  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable as well.  $\square$

**Corollary 2.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X}) - \mathcal{X}$  be a nonvariable term and  $\sigma$  be a substitution such that  $\sigma(t) \in \mathcal{M}_{\infty, \mu}$ . Then,  $\text{REN}^\mu(t)$  is  $\mu$ -narrowable.

**Proof.** By Proposition 5, there is a rule  $l \rightarrow r$  and a substitution  $\sigma$  such that  $\sigma(t) \xrightarrow{>\Delta}^*_{\mathcal{R}, \mu} \sigma(l)$  (since we can assume that variables in  $l$  and variables in  $t$  are disjoint, we can apply the same substitution  $\sigma$  to  $t$  and  $l$  without any problem). By Proposition 6, the conclusion follows.  $\square$

In the following, we write  $\text{NARR}_{\mathcal{R}}^\mu(t)$  (or just  $\text{NARR}^\mu(t)$ ) to indicate that  $t$  is  $\mu$ -narrowable with respect to the (intended) TRS  $\mathcal{R}$ . We also let

$$\mathcal{NH}(\mathcal{R}, \mu) = \{t \in \mathcal{DH}(\mathcal{R}, \mu) \mid \text{NARR}_{\mathcal{R}}^\mu(\text{REN}^\mu(t))\}$$

be the set of *hidden terms* that are rooted by a *defined* symbol, and that after applying  $\text{REN}^\mu$  become  $\mu$ -narrowable.

**Example 9.** Since all terms  $t \in \mathcal{DHT}(\mathcal{R}, \mu)$  for  $\mathcal{R}$  and  $\mu$  as in Example 8 are  $\mu$ -narrowable (even without applying  $\text{REN}^\mu$ ), we have  $\mathcal{NHT}(\mathcal{R}, \mu) = \mathcal{DHT}(\mathcal{R}, \mu)$ .

As a consequence of the previous results, we have the following main result, which we use later.

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

$$t = t_0 \xrightarrow{>\Lambda}^* \sigma_1(l_1) \xrightarrow{\Lambda} \sigma_1(r_1) \succeq_{\mu} t_1 \xrightarrow{>\Lambda}^* \sigma_2(l_2) \xrightarrow{\Lambda} \sigma_2(r_2) \succeq_{\mu} t_2 \xrightarrow{>\Lambda}^* \dots$$

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

1.  $t_i = \sigma_i(s_i)$  for some nonvariable term  $s_i$  such that  $r_i \succeq_{\mu} s_i$ , or
2.  $\sigma_i(x_i) \succeq_{\mu} t_i$  for some  $x_i \in \text{Var}^\mu(r_i) - \text{Var}^\mu(l_i)$  and  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{NHT}$  and substitution  $\theta_i$ .

**Proof.** Since  $\mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ , by Corollary 1, we have a sequence

$$t = t_0 \xrightarrow{>\Lambda}^* \sigma_1(l_1) \xrightarrow{\Lambda} \sigma_1(r_1) \succeq_{\mu} t_1 \xrightarrow{>\Lambda}^* \sigma_2(l_2) \xrightarrow{\Lambda} \sigma_2(r_2) \succeq_{\mu} t_2 \xrightarrow{>\Lambda}^* \dots$$

where, for all  $i \geq 1$ ,  $l_i \rightarrow r_i \in \mathcal{R}$ ,  $\sigma_i$  are substitutions,  $t_i \in \mathcal{M}_{\infty, \mu}$ , and either (1)  $t_i = \sigma_i(s_i)$  for some nonvariable term  $s_i$  such that  $r_i \succeq_{\mu} s_i$  or (2)  $\sigma_i(x_i) \succeq_{\mu} t_i$  for some  $x_i \in \text{Var}^\mu(r_i) - \text{Var}^\mu(l_i)$  (and hence  $\sigma(l_i) \triangleright_{\mu} t_i$  and  $\sigma(r_i) \succeq_{\mu} t_i$  as well). We only need to prove that terms  $t_i$  are instances of hidden terms in  $\mathcal{NHT}$  whenever (2) holds. By Proposition 4, for all such terms  $t_i$ , we have that either (A)  $\sigma_1(l_1) \triangleright_{\mu} t_i$  or (B)  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{DHT}$  and substitution  $\theta_i$ . In case (B), we just

consider Corollary 2, which ensures that  $t'_i \in \mathcal{NHT}$ . In case (A), since  $t \xrightarrow{>\Lambda}^* \sigma_1(l_1)$  and  $\sigma_1(l_1)$  is not  $\mu$ -terminating, by Lemma 4, all terms  $u_j$  in the  $\mu$ -rewrite sequence

$$t = u_1 \xrightarrow{>\Lambda} u_2 \xrightarrow{>\Lambda} \dots \xrightarrow{>\Lambda} u_m = \sigma_1(l_1)$$

belong to  $\mathcal{M}_{\infty, \mu}$ :  $u_j \in \mathcal{M}_{\infty, \mu}$  for all  $j$ ,  $1 \leq j \leq m$ . Since  $t \in \mathcal{T}_{\infty, \mu}$ , all its strict subterms (disregarding their  $\mu$ -replacing character) are  $\mu$ -terminating. Since  $t_i$  is not  $\mu$ -terminating,  $t \not\triangleright_{\mu} t_i$ . By Lemma 6,  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{DHT}$  and substitution  $\theta_i$ . By Corollary 2,  $t'_i \in \mathcal{NHT}$ .  $\square$

#### 4. Context-sensitive dependency pairs

By Lemma 2 every non- $\mu$ -terminating term  $s_0$  contains a strongly minimal subterm  $t \in \mathcal{T}_{\infty, \mu}$  which, by Theorem 1, starts an infinite  $\mu$ -rewrite sequence. In such a sequence, a number of  $\mu$ -rewriting steps *below the root* of  $t$  are performed. Then a rule  $l \rightarrow r$  is applied at the *topmost* position of the obtained reduct. According to Proposition 5, the application of such a rule either

1. *introduces* a new minimal non- $\mu$ -terminating subterm  $u$  having a prefix  $s$  which is a nonvariable  $\mu$ -replacing subterm of  $r$ . By Corollary 2,  $\text{REN}^\mu(s)$  is  $\mu$ -narrowable. Otherwise,
2. *takes* a minimal non- $\mu$ -terminating and non- $\mu$ -replacing subterm  $u$  and
  - (a) brings it up to an *active* position by means of the binding  $\sigma(x)$  for some *migrating variable*  $x$  in  $l \rightarrow r$ .
  - (b) At this point, we know that  $u$ , which is rooted by a defined symbol due to  $u \in \mathcal{M}_{\infty, \mu}$ , is an instance of a hidden term  $u' \in \mathcal{NHT}$ .

Afterwards, further *inner*  $\mu$ -rewritings on  $u$  lead to a matching with the left-hand-side  $l'$  of a new rule  $l' \rightarrow r'$  and everything starts again. This process is abstracted in the definition of *context-sensitive dependency pairs* and in the definition of chain below.

Given a signature  $\mathcal{F}$  and  $f \in \mathcal{F}$ , we let  $f^\sharp$  be a new fresh symbol (often called *tuple symbol* or DP-symbol) associated to a symbol  $f$  [10]. Let  $\mathcal{F}^\sharp$  be the set of tuple symbols associated to symbols in  $\mathcal{F}$ . As usual, for  $t = f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^\sharp$  to denote the *marked term*  $f^\sharp(t_1, \dots, t_k)$ . Conversely, given a marked term  $t = f^\sharp(t_1, \dots, t_k)$ , where  $t_1, \dots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we write  $t^\natural$  to denote the term  $f(t_1, \dots, t_k) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ .

**Definition 4** (*Context-sensitive dependency pairs*). Let  $\mathcal{R} = (\mathcal{F}, R) = (C \uplus D, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  be the set of *context-sensitive dependency pairs* (CSDPs) where:

$$\begin{aligned} \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) &= \left\{ l^{\sharp} \rightarrow s^{\sharp} \mid l \rightarrow r \in \mathcal{R}, r \triangleright_{\mu} s, \text{root}(s) \in \mathcal{D}, l \not\triangleright_{\mu} s, \text{NARR}^{\mu}(\text{REN}^{\mu}(s)) \right\} \\ \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) &= \left\{ l^{\sharp} \rightarrow x \mid l \rightarrow r \in \mathcal{R}, x \in \text{Var}^{\mu}(r) - \text{Var}^{\mu}(l) \right\} \end{aligned}$$

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

The CSDPs  $u \rightarrow v \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  in Definition 4, consisting of collapsing rules only, are called the *collapsing* CSDPs.

**Remark 3.** The notion of CSDP in Definition 4 differs from the standard definition of dependency pair [10,35] in two additional requirements:

1. As in [38], which follows Dershowitz's proposal in [15], we require that subterms  $s$  of the right-hand sides  $r$  of the rules  $l \rightarrow r$  which are considered to build the dependency pairs  $l^{\sharp} \rightarrow s^{\sharp}$  are not subterms of the left-hand side (i.e.,  $l \not\triangleright_{\mu} s$ ).
2. As in [53], we require  $\mu$ -narrowability of  $\text{REN}^{\mu}(s)$ :  $\text{NARR}^{\mu}(\text{REN}^{\mu}(s))$ .

But the crucial difference, which is specific for context-sensitive rewriting, is the introduction and use of *collapsing* dependency pairs.

A rule  $l \rightarrow r$  of a TRS  $\mathcal{R}$  is  $\mu$ -conservative if  $\text{Var}^{\mu}(r) \subseteq \text{Var}^{\mu}(l)$ , i.e., there is no migrating variable;  $\mathcal{R}$  is  $\mu$ -conservative if all its rules are  $\mu$ -conservative (see [43,50]). The following fact is obvious from Definition 4.

**Proposition 7.** *If  $\mathcal{R}$  is a  $\mu$ -conservative TRS, then  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ .*

Therefore, in order to deal with  $\mu$ -conservative TRSs  $\mathcal{R}$  we only need to consider the 'classical' dependency pairs in  $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ .

$$\begin{aligned} \text{ADD}(s(n), m) &\rightarrow \text{ADD}(n, m) & (1) \\ \text{HALFPI}(n) &\rightarrow \text{EVENNS} & (4) \\ \text{HALFPI}(n) &\rightarrow \text{ODDNS} & (5) \\ \text{HALFPI}(n) &\rightarrow \text{PRODOFFRACS}(\text{take}(n, \text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs})))) & (6) \\ \text{HALFPI}(n) &\rightarrow \text{REP2}(\text{oddNs}) & (7) \\ \text{HALFPI}(n) &\rightarrow \text{REP2}(\text{tail}(\text{evenNs})) & (8) \\ \text{HALFPI}(n) &\rightarrow \text{TAIL}(\text{evenNs}) & (9) \\ \text{HALFPI}(n) &\rightarrow \text{TAIL}(\text{rep2}(\text{oddNs})) & (10) \\ \text{HALFPI}(n) &\rightarrow \text{TAKE}(n, \text{zip}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs})))) & (11) \\ \text{HALFPI}(n) &\rightarrow \text{ZIP}(\text{rep2}(\text{tail}(\text{evenNs})), \text{tail}(\text{rep2}(\text{oddNs}))) & (12) \\ \text{ODDNS} &\rightarrow \text{EVENNS} & (14) \\ \text{ODDNS} &\rightarrow \text{INCR}(\text{evenNs}) & (15) \\ \text{PROD}(s(n), m) &\rightarrow \text{ADD}(m, \text{prod}(n, m)) & (16) \\ \text{PROD}(s(n), m) &\rightarrow \text{PROD}(n, m) & (17) \\ \text{PRODFRAC}(\text{frac}(x, y), \text{frac}(z, t)) &\rightarrow \text{PROD}(x, z) & (18) \\ \text{PRODFRAC}(\text{frac}(x, y), \text{frac}(z, t)) &\rightarrow \text{PROD}(y, t) & (19) \\ \text{PRODOFFRACS}(\text{consF}(p, ps)) &\rightarrow \text{PRODFRAC}(p, \text{prodOfFrac}(ps)) & (20) \\ \text{PRODOFFRACS}(\text{consF}(p, ps)) &\rightarrow \text{PRODOFFRACS}(ps) & (21) \\ \text{TAKE}(s(n), \text{cons}(x, xs)) &\rightarrow \text{TAKE}(n, xs) & (23) \\ \text{TAIL}(\text{cons}(x, xs)) &\rightarrow xs & (25) \\ \text{TAKE}(s(n), \text{cons}(x, xs)) &\rightarrow xs & (26) \end{aligned}$$

Fig. 4. Context-sensitive dependency pairs for the CS-TRS in Example 1.

**Example 10.** Consider the following TRS  $\mathcal{R}$ :

$$\begin{array}{ll} g(x) \rightarrow h(x) & h(d) \rightarrow g(c) \\ c \rightarrow d & \end{array}$$

together with  $\mu(g) = \mu(h) = \emptyset$  [63, Example 1]. Note that  $\mathcal{R}$  is  $\mu$ -conservative.  $\text{DP}(\mathcal{R}, \mu)$  consists of the following (noncollapsing) CSDPs:

$$\begin{array}{ll} G(x) \rightarrow H(x) & H(d) \rightarrow G(c) \end{array}$$

with  $\mu^\sharp(G) = \mu^\sharp(H) = \emptyset$ .

If the TRS  $\mathcal{R}$  contains non- $\mu$ -conservative rules, then we also need to consider dependency pairs with variables in the right-hand side.

**Example 11.** As discussed in Examples 2 and 4, for the CS-TRS  $(\mathcal{R}, \mu)$  in Example 1, we have the CSDPs in Fig. 4.

### 5. Chains of CSDPs

An essential property of the dependency pair method is that it provides a *characterization* of termination of TRSs  $\mathcal{R}$  as the absence of infinite (minimal) *chains of dependency pairs* [10, 35]. As we prove in Section 6, this is also true for CSR when CSDPs are considered. First, we have to introduce a suitable notion of chain that can be used with CSDPs. As in the DP-framework [33, 35], where the origin of *pairs* does not matter, we use another TRS  $\mathcal{P}$  together with  $\mathcal{R}$  to build the chains. Once this more abstract notion of chain is introduced, it can be particularized to be used with CSDPs, by just taking  $\mathcal{P} = \text{DP}(\mathcal{R}, \mu)$ .

**Definition 5** (*Chain of pairs – minimal chain*). Let  $\mathcal{R} = (F, R)$  and  $\mathcal{P} = (G, P)$  be TRSs and  $\mu \in M_{F \cup G}$ . A  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain is a finite or infinite sequence of pairs  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma : \mathcal{X} \rightarrow \mathcal{T}(F \cup G, \mathcal{X})$  satisfying that, for all  $i \geq 1$ :

1. if  $v_i \notin \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_{i+1})$ , and
2. if  $v_i \in \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) = C_i[s_i]_{p_i}$  for some  $s_i$  and  $C_i[\ ]_{p_i}$  such that  $p_i \in \text{Pos}^\mu(C_i[\ ]_{p_i})$ ,  $\text{prefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ , and  $s_i \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_{i+1})$ .

As usual, we assume that different occurrences of pairs do not share any variable (renaming substitutions are used if necessary). A  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain is called *minimal* if for all  $i \geq 1$ ,

1. if  $v_i \notin \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $\sigma(v_i)$  is  $(\mathcal{R}, \mu)$ -terminating, and
2. if  $v_i \in \text{Var}(u_i) - \text{Var}^\mu(u_i)$ , then  $s_i^\sharp$  is  $(\mathcal{R}, \mu)$ -terminating and  $\exists \bar{s}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$  such that  $s_i = \sigma(\bar{s}_i)$ .

Note that the condition  $v_i \in \text{Var}(u_i) - \text{Var}^\mu(u_i)$  in Definition 5 implies that  $v_i$  is a variable. Furthermore,  $v_i$  is a *migrating variable* in the rule  $u_i \rightarrow v_i$ .

**Remark 4** (*Conventions about  $\mathcal{P}$* ). The following conventions about the component  $\mathcal{P} = (G, P)$  of our chains will be observed during our development:

1. According to the usual terminology [35], we often call *pairs* the rules  $u \rightarrow v \in \mathcal{P}$ .
2. We have to mark terms  $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  before *connecting* them to the instance  $\sigma(u_{i+1})$  of the left-hand side of the next pair. Since marked symbols  $f^\sharp$  are *fresh* (with respect to the signature  $\mathcal{F}$  of the TRS  $\mathcal{R}$ ), we also assume that  $\mathcal{D}^\sharp \cap \mathcal{F} = \emptyset$  and  $\mathcal{D}^\sharp \subseteq \mathcal{G}$ .
3. We assume that  $\mathcal{P}$  contains a *finite* set of rules. This is essential in many proofs.

In the following, the pairs in a CS-TRS  $(\mathcal{P}, \mu)$ , where  $\mathcal{P} = (G, P)$ , are partitioned according to their role in Definition 5 as follows:

$$P_{\mathcal{X}} = \{u \rightarrow v \in P \mid v \in \text{Var}(u) - \text{Var}^\mu(u)\} \text{ and } P_{\mathcal{G}} = P - P_{\mathcal{X}}$$

**Remark 5** (*Collapsing pairs*). Note that all pairs in  $P_{\mathcal{X}} = (G, P_{\mathcal{X}})$  are *collapsing*. The rules in  $P_{\mathcal{G}} = (G, P_{\mathcal{G}})$  can be *collapsing* as well: a rewrite rule  $f(x) \rightarrow x \in \mathcal{P}$  with  $\mu(f) = \{1\}$  does *not* belong to  $P_{\mathcal{X}}$  but rather to  $P_{\mathcal{G}}$  because  $x$  is not a migrating variable.

Despite this fact, we refer to  $\mathcal{P}_X$  as the set of *collapsing* pairs in  $\mathcal{P}$  because its intended role in Definition 5 is capturing the computational behavior of collapsing CSDPs in  $\text{DP}_X(\mathcal{R}, \mu)$ .

**Remark 6** (Notation for chains). In general, a  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain can be written as follows:

$$\sigma(u_1) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^{\#} t_1 \xrightarrow{\mathcal{R}, \mu} \sigma(u_2) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^{\#} t_2 \xrightarrow{\mathcal{R}, \mu} \dots$$

where, for all  $i \geq 1$  and  $u_i \rightarrow v_i \in \mathcal{P}$ ,

1. if  $u_i \rightarrow v_i \notin \mathcal{P}_X$ , then  $t_i = \sigma(v_i)$ ,
2. if  $u_i \rightarrow v_i \in \mathcal{P}_X$ , then  $t_i = s_i^{\#}$  for some term  $s_i$  such that  $\sigma(v_i) = C_i[s_i]_{p_i}$  for some  $C_i[\ ]_{p_i}$  such that  $p_i \in \text{Pos}^{\mu}(C_i[\ ]_{p_i})$ , and  $\text{spre}^{\mu}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ .

This is denoted in a compact way by  $\sigma(u_i) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^{\#} t_i$  emphasizing that there is a  $\mathcal{P}$ -step followed by either an equality step (as in (1)) or by  $\mu$ -replacing projection steps (restricted to symbols in  $\mathcal{F}$ ) plus a marking operation (as in (2)) depending on the considered pair  $u_i \rightarrow v_i$ .

### 5.1. Properties of some particular chains

In the following, we let  $\mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu) \subseteq \mathcal{NHT}(\mathcal{R}, \mu)$  (or just  $\mathcal{NHT}_{\mathcal{P}}$ ) be as follows:

$$\mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu) = \left\{ t \in \mathcal{NHT}(\mathcal{R}, \mu) \mid \exists u \rightarrow v \in \mathcal{P}, \exists \theta, \theta', \theta(t^{\#}) \xrightarrow{\mathcal{R}, \mu} \theta'(u) \right\}$$

This set contains the narrowable hidden terms that ‘connect’ with pairs in  $\mathcal{P}$ .

**Remark 7.** Note that  $\mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu)$  is not computable, in general, due to the need for checking the reachability of  $\theta'(u)$  from  $\theta(t^{\#})$  using CSR. Suitable (over)approximations are discussed below (see Remark 10).

We let  $\mathcal{P}_X^1$  denote the subTRS of  $\mathcal{P}_X$  containing the rules whose migrating variables occur on non- $\mu$ -replacing immediate subterms in the left-hand side:

$$\mathcal{P}_X^1 = \{ f(u_1, \dots, u_k) \rightarrow x \in \mathcal{P}_X \mid \exists i, 1 \leq i \leq k, i \notin \mu(f), x \in \text{Var}(u_i) \}$$

**Proposition 8.** Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ .

1. If  $\mathcal{NHT}_{\mathcal{P}} = \emptyset$ , then every infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain is an infinite minimal  $(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)$ -chain and there is no infinite minimal  $(\mathcal{P}_X, \mathcal{R}, \mu)$ -chain.
2. If  $\mathcal{P} = \mathcal{P}_X^1$ , then there is no infinite  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain.

**Proof.**

1. By contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain containing any  $u_i \rightarrow v_i \in \mathcal{P}_X$ . By Definition 5, such a pair must be followed by a pair  $u_{i+1} \rightarrow v_{i+1} \in \mathcal{P}$  such that  $\theta_i(\bar{s}_i^{\#}) \xrightarrow{\mathcal{R}, \mu} \sigma(u_{i+1})$  for some  $\bar{s}_i \in \mathcal{NHT}$  and substitution  $\theta_i$ . Therefore,  $t_i^{\#} \in \mathcal{NHT}_{\mathcal{P}}$ , but  $\mathcal{NHT}_{\mathcal{P}} = \emptyset$ , leading to a contradiction.
2. By contradiction. Assume that there is an infinite chain that only uses dependency pairs  $u_i \rightarrow x_i \in \mathcal{P}_X^1$  for all  $i \geq 1$ . Let  $f_i = \text{root}(u_i)$  for  $i \geq 1$ . Then, by definition of  $\mathcal{P}_X^1$ , for all  $i \geq 1$ , there is  $j_i \in \{1, \dots, \text{ar}(f_i)\} - \mu(f_i)$  such that  $u_i|_{j_i} \triangleright x_i$ . According to Definition 5, we have that  $\sigma(u_i)|_{j_i} \triangleright \sigma(x_i) \triangleright_{\mu} s_i$  for some term  $s_i$  such that  $s_i^{\#} \xrightarrow{\mathcal{R}, \mu} \sigma(u_{i+1})$ . Since  $\text{root}(s_i^{\#}) \in \mathcal{D}^{\#} \subseteq \mathcal{G}$  and  $\mathcal{D}^{\#} \cap \mathcal{F} = \emptyset$  (Remark 4), no  $\mu$ -rewriting step is possible at the root of  $s_i^{\#}$ . Thus,  $\text{root}(s_i^{\#}) = \text{root}(u_{i+1}) = f_{i+1}$  and  $j_{i+1} \notin \mu(f_{i+1})$ . Since no  $\mu$ -rewriting step is possible on the  $j_{i+1}$ th immediate subterm  $s_i^{\#}|_{j_{i+1}}$  of  $s_i^{\#}$ , it follows that  $s_i^{\#}|_{j_{i+1}} = \sigma(u_{i+1})|_{j_{i+1}} \triangleright \sigma(x_{i+1})$ , i.e.,  $\sigma(x_i) \triangleright \sigma(x_{i+1})$  for all  $i \geq 1$ . We get an infinite sequence  $\sigma(x_1) \triangleright \sigma(x_2) \triangleright \dots$  which contradicts well-foundedness of  $\triangleright$ .  $\square$

The following proposition establishes some important ‘basic’ cases of (absence of) infinite context-sensitive chains of pairs which are used later.

**Proposition 9.** Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ .

1. If  $P = \emptyset$ , then every  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain is empty.

2. If  $R = \emptyset$ , then there is no infinite  $(\mathcal{P}_X, \mathcal{R}, \mu)$ -chain.
3. Let  $u \rightarrow v \in \mathcal{P}_G$  be such that  $v = \theta(u)$ . Then, there is an infinite  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain.

**Proof.**

1. Obvious, by Definition 5.
2. By contradiction. If there is an infinite  $(\mathcal{P}_X, \mathcal{R}, \mu)$ -chain, then, since there is no rule in  $\mathcal{R}$ , there is a substitution  $\sigma$  such that

$$\sigma(u_1) \hookrightarrow_{\mathcal{P}, \mu} \sigma(x_1) \triangleright_{\mu}^{\#} t_1 = \sigma(u_2) \hookrightarrow_{\mathcal{P}, \mu} \sigma(x_2) \triangleright_{\mu}^{\#} t_2 = \sigma(u_3) \dots$$

where  $t_i = s_i^{\#}$  for some terms  $s_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) = C_i[s_i]_{p_i}$  for some  $C_i[\ ]_{p_i}$  and  $p_i \in \text{Pos}^{\mu}(C_i[\ ]_{p_i})$  such that  $\text{sprex}(p_i) \subseteq \mathcal{F}$  for  $i \geq 1$ . Since  $x_i \in \text{Var}(u_i)$  and  $u_i$  is not a variable, we have  $u_i \triangleright x_i$ ; hence,  $\sigma(u_i) \triangleright \sigma(x_i)$  (by stability of  $\triangleright$ ) and also  $\sigma(u_i) \triangleright s_i$  for all  $i \geq 1$ . Since  $s_i$  and  $\sigma(u_{i+1})$  only differ in the root symbol, we can actually say that  $s_i \triangleright s_{i+1}$  for all  $i \geq 1$ . Thus, we obtain an infinite sequence  $s_1 \triangleright s_2 \triangleright \dots$  that contradicts the well-foundedness of  $\triangleright$ .

3. Trivial.  $\square$

The following example shows that Proposition 9(2) does not hold for TRSs  $\mathcal{P}$  with arbitrary rules.

**Example 12.** Consider  $\mathcal{P} = \{\mathbb{F}(x) \rightarrow x, \mathbb{G}(x) \rightarrow \mathbb{F}(\mathbb{G}(x))\}$  together with a TRS  $\mathcal{R}$  with an empty set of rules:  $\mathcal{R} = (\{\mathbb{G}\}, \emptyset)$ . Let  $\mu$  be given by  $\mu(f) = \emptyset$  for all  $f \in \mathcal{F} \cup \mathcal{G}$ . Note that  $\mathcal{P}_X$  consists of the pair  $\mathbb{F}(x) \rightarrow x$  because  $x \in \text{Var}(\mathbb{F}(x)) - \text{Var}^{\mu}(\mathbb{F}(x))$ . Then, we have an infinite chain

$$\mathbb{F}(\mathbb{G}(x)) \hookrightarrow_{\mathcal{P}, \mu} \mathbb{G}(x) \triangleright_{\mu}^{\#} \mathbb{G}(x) \hookrightarrow_{\mathcal{P}, \mu} \mathbb{F}(\mathbb{G}(x)) \hookrightarrow_{\mathcal{R}, \mu} \dots$$

Since  $\mathcal{NHT} = \emptyset$ ,  $\mathbb{G}(x)$  is not an instance of any term in  $\mathcal{NHT}$ . Thus, the chain is *not* minimal.

### 5.2. Chains of CSDPs vs. chains of DPs

The definition of chain of CSDPs differs from the one for DPs. First, we use  $\hookrightarrow^*$  instead of  $\rightarrow^*$  for connecting pairs. Also, we require  $\mu$ -termination instead of termination for minimal chains. However, the most important difference concerns the treatment of collapsing pairs. In general (and in sharp contrast with the DP approach), the connection between the right-hand side of a collapsing pair (which is a variable, e.g.,  $x$ ) and the left-hand side  $u$  of the next pair in the chain depends on whether a marked *narrowable hidden term* (which is introduced by a *previous*  $\mu$ -rewriting step)  $\mu$ -rewrites into  $\sigma(u)$ . Dealing with collapsing pairs, hidden terms can be thought of as playing the role of *hidden* or *delayed* recursive paths. This fits the guiding idea of the DP approach as an analysis of rewriting-based recursion paths in function calls (as briefly discussed in Section 1).

## 6. Characterizing termination of CSR using chains of CSDPs

The following result establishes the soundness of the CSDP approach.

**Theorem 2** (Soundness). *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if there is no infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\#})$ -chain.*

**Proof.** By contradiction. If  $\mathcal{R}$  is not  $\mu$ -terminating, then there is  $t \in \mathcal{T}_{\infty, \mu}$  (Lemma 2). By Theorem 1, there are rules  $l_i \rightarrow r_i \in \mathcal{R}$ , substitutions  $\sigma_i$ , and terms  $t_i \in \mathcal{M}_{\infty, \mu}$ , for  $i \geq 1$  such that

$$t = t_0 \xrightarrow{>^{\Lambda}}^* \sigma_1(l_1) \xrightarrow{\Lambda} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{>^{\Lambda}}^* \sigma_2(l_2) \xrightarrow{\Lambda} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{>^{\Lambda}}^* \dots$$

where either (D1)  $t_i = \sigma_i(s_i)$  for some  $s_i$  such that  $r_i \triangleright_{\mu} s_i$  or (D2)  $\sigma_i(x_i) \triangleright_{\mu} t_i$  for some  $x_i \in \text{Var}^{\mu}(r_i) - \text{Var}^{\mu}(l_i)$  and  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{NHT}$ . Furthermore, since  $t_{i-1} \xrightarrow{>^{\Lambda}}^* \sigma_i(l_i)$  and  $t_{i-1} \in \mathcal{M}_{\infty, \mu}$  (in particular,  $t_0 = t \in \mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ ), by Lemma 4,  $\sigma_i(l_i) \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . Note that, since  $t_i \in \mathcal{M}_{\infty, \mu}$ , we have that  $t_i^{\#}$  is  $\mu$ -terminating (with respect to  $\mathcal{R}$ ), because all  $\mu$ -replacing subterms of  $t_i$  (hence of  $t_i^{\#}$  as well) are  $\mu$ -terminating and  $\text{root}(t_i^{\#})$  is not a defined symbol of  $\mathcal{R}$ .

First, note that  $\text{DP}(\mathcal{R}, \mu)$  is a TRS  $\mathcal{P}$  over the signature  $\mathcal{G} = \mathcal{F} \cup \mathcal{D}^{\#}$  and  $\mu^{\#} \in M_{\mathcal{F} \cup \mathcal{G}}$  as required by Definition 5. Furthermore,  $\mathcal{P}_{\mathcal{G}} = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  and  $\mathcal{P}_X = \text{DP}_X(\mathcal{R}, \mu)$ . We can define an infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\#})$ -chain using CSDPs  $u_i \rightarrow v_i$  for  $i \geq 1$ , where  $u_i = l_i^{\#}$  and

1.  $v_i = s_i^\sharp$  if (D1) holds. Since  $t_i \in \mathcal{M}_{\infty, \mu}$ , we have that  $\text{root}(s_i) \in \mathcal{D}$  and, because  $t_i = \sigma_i(s_i)$ , by Corollary 2,  $\text{REN}^\mu(s_i)$  is  $\mu$ -narrowable. Furthermore, if we assume that  $s_i$  is a  $\mu$ -replacing subterm of  $l_i$  (i.e.,  $l_i \triangleright_\mu s_i$ ), then  $\sigma_i(l_i) \triangleright_\mu \sigma_i(s_i)$ . Since  $\sigma_i(s_i) = t_i \in \mathcal{M}_{\infty, \mu}$ , this contradicts that  $\sigma_i(l_i) \in \mathcal{M}_{\infty, \mu}$ . Thus,  $l_i \not\triangleright_\mu s_i$ . Hence,  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ .  
Furthermore,  $t_i^\sharp = \sigma_i(v_i)$  is  $\mu$ -terminating. Finally, since  $t_i = \sigma_i(s_i) \xrightarrow{> \Delta} \sigma_{i+1}(l_{i+1})$  and  $\mu^\sharp$  extends  $\mu$  to  $\mathcal{F} \cup \mathcal{D}^\sharp$  by  $\mu^\sharp(f^\sharp) = \mu(f)$  for all  $f \in \mathcal{D}$ , we also have that  $\sigma_i(v_i) = \sigma_i(s_i^\sharp) \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma_{i+1}(u_{i+1})$ .
2.  $v_i = x_i$  if (D2) holds. Clearly,  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . As discussed above,  $t_i^\sharp$  is  $\mu$ -terminating. Since  $\sigma_i(x_i) \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $\sigma_i(x_i) \triangleright_\mu t_i$ , we have that  $\sigma(v_i) = C_i[t_i]_{p_i}$  for some  $C_i[\ ]_{p_i}$  and  $p_i \in \mathcal{Pos}^\mu(C_i[\ ]_{p_i})$  such that  $\text{prefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ .  
Finally, since  $t_i \xrightarrow{> \Delta} \sigma_{i+1}(l_{i+1})$ , again we have that  $t_i^\sharp \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma_{i+1}(u_{i+1})$ . Furthermore,  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{NHT}$  and substitution  $\theta_i$ .

Regarding  $\sigma$ , w.l.o.g. we can assume that  $\text{Var}(l_i) \cap \text{Var}(l_j) = \emptyset$  for all  $i \neq j$ , and therefore  $\text{Var}(u_i) \cap \text{Var}(u_j) = \emptyset$  as well. Then,  $\sigma$  is given by  $\sigma(x) = \sigma_i(x)$  whenever  $x \in \text{Var}(u_i)$  for  $i \geq 1$ . From the discussion in (1) and (2), we conclude that the CSDPs  $u_i \rightarrow v_i$  together with  $\sigma$  define an infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain. This leads to a contradiction.  $\square$

Let  $\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu) = \mathcal{P}_{\mathcal{X}}^1$  for  $\mathcal{P} = \text{DP}(\mathcal{R}, \mu)$ . By Theorem 2 and Propositions 8 and 9, we have the following.

**Corollary 3** (Basic  $\mu$ -termination criteria). *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ .*

1. If  $\text{DP}(\mathcal{R}, \mu) = \emptyset$ , then  $\mathcal{R}$  is  $\mu$ -terminating.
2. If  $\mathcal{NHT}_{\text{DP}(\mathcal{R}, \mu)}(\mathcal{R}, \mu) = \emptyset$  and  $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) = \emptyset$ , then  $\mathcal{R}$  is  $\mu$ -terminating.
3. If  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$ , then  $\mathcal{R}$  is  $\mu$ -terminating.

**Example 13.** Consider the following TRS  $\mathcal{R}$  [44, Example 15]:

$$\begin{array}{ll}
 \text{and}(\text{true}, x) \rightarrow x & \text{add}(0, x) \rightarrow x \\
 \text{and}(\text{false}, y) \rightarrow \text{false} & \text{add}(s(x), y) \rightarrow s(\text{add}(x, y)) \\
 \text{if}(\text{true}, x, y) \rightarrow x & \text{from}(x) \rightarrow \text{cons}(x, \text{from}(s(x))) \\
 \text{if}(\text{false}, x, y) \rightarrow y & \text{first}(0, x) \rightarrow \text{nil} \\
 \text{first}(s(x), \text{cons}(y, z)) \rightarrow \text{cons}(y, \text{first}(x, z)) & 
 \end{array}$$

together with the canonical replacement map  $\mu(\text{cons}) = \mu(s) = \mu(\text{from}) = \emptyset$ ,  $\mu(\text{add}) = \mu(\text{and}) = \mu(\text{if}) = \{1\}$ , and  $\mu(\text{first}) = \{1, 2\}$ , which ensures completeness of CSR for computing head-normal forms<sup>3</sup> with  $\mathcal{R}$  (see [44, 46]). Then,  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$  is:

$$\begin{array}{ll}
 \text{AND}(\text{true}, x) \rightarrow x & \text{IF}(\text{true}, x, y) \rightarrow x \\
 \text{ADD}(0, x) \rightarrow x & \text{IF}(\text{false}, x, y) \rightarrow y
 \end{array}$$

Note also that  $\mathcal{NHT}_{\text{DP}(\mathcal{R}, \mu)} = \emptyset$ . Thus, by either of the last two statements of Corollary 3, we conclude the  $\mu$ -termination of  $\mathcal{R}$ .

The following example shows that Corollary 3(3) does not hold for chains consisting of arbitrary collapsing CSDPs.

**Example 14.** Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 3. Note that  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  (both  $\text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  and  $\text{DP}_{\mathcal{X}}^1(\mathcal{R}, \mu)$  are empty!). We have the following infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain:

$$F(a) \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} F(c(F(a))) \xrightarrow{*}_{\text{DP}(\mathcal{R}, \mu), \mu^\sharp} F(a) \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \dots$$

Now, we prove that the previous CS-dependency pair approach is not only correct but also complete for proving termination of CSR.

**Theorem 3** (Completeness). *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If  $\mathcal{R}$  is  $\mu$ -terminating, then there is no infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain.*

<sup>3</sup> A head-normal form is a term that cannot be rewritten to a redex.

**Proof.** By contradiction. If there is an infinite  $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain, then there are a substitution  $\sigma$  and dependency pairs  $u_i \rightarrow v_i \in DP(\mathcal{R}, \mu)$  such that

1.  $\sigma(v_i) \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
2. if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is  $s_i \in T(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) \succeq_{\mu} s_i$  and  $s_i^\sharp \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})$ .

for  $i \geq 1$ . Now, consider the first dependency pair  $u_1 \rightarrow v_1$  in the sequence:

1. If  $u_1 \rightarrow v_1 \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$ , then  $v_1^\sharp$  is a  $\mu$ -replacing subterm of the right-hand-side  $r_1$  of a rule  $l_1 \rightarrow r_1$  in  $\mathcal{R}$ . Therefore,  $r_1 = C_1[v_1^\sharp]_{p_1}$  for some position  $p_1 \in Pos^\mu(r_1)$  and context  $C_1[\ ]_{p_1}$ , and we can perform the  $\mu$ -rewriting step  $t_1 = \sigma(u_1) \xrightarrow{\mathcal{R}, \mu} \sigma(r_1) = \sigma(C_1[\sigma(v_1^\sharp)]_{p_1}) = s_1$ , where  $\sigma(v_1^\sharp) = \sigma(v_1) \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma(u_2)$  and  $\sigma(u_2)$  initiates an infinite  $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain. Note that  $p_1 \in Pos^\mu(s_1)$ .
2. If  $u_1 \rightarrow x \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is a rule  $l_1 \rightarrow r_1$  in  $\mathcal{R}$  such that  $u_1 = l_1^\sharp$ , and  $x \in Var^\mu(r_1) - Var^\mu(l_1)$ , i.e.,  $r_1 = C_1[x]_{q_1}$  for some  $q_1 \in Pos^\mu(r_1)$ . Furthermore, since  $\sigma(x) = C'_1[s]_{p'_1}$  for some term  $s$ ,  $C'_1[\ ]_{p'_1}$  and  $p'_1 \in Pos^\mu(C'_1[\ ]_{p'_1})$  such that  $s^\sharp \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma(u_2)$ , we can perform the  $\mu$ -rewriting step  $t_1 = \sigma(l_1) \xrightarrow{\mathcal{R}, \mu} \sigma(r_1) = \sigma(C_1[C'_1[s]_{p'_1}]_{q_1}) = s_1$  where  $s^\sharp \xrightarrow{*}_{\mathcal{R}, \mu^\sharp} \sigma(u_2)$  (hence  $s \xrightarrow{> \Lambda}_{\mathcal{R}, \mu} u_2^\sharp$ ) and  $\sigma(u_2)$  initiates an infinite  $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain. Note that  $p_1 = q_1 \cdot p'_1 \in Pos^\mu(s_1)$  (use Proposition 1).

Since  $\mu^\sharp(f^\sharp) = \mu(f)$ , and  $p_1 \in Pos^\mu(s_1)$ , we have that  $s_1 \xrightarrow{*}_{\mathcal{R}, \mu} t_2[\sigma(u_2)]_{p_1} = t_2$  and  $p_1 \in Pos^\mu(t_2)$ . Thus, we can build an infinite  $\mu$ -rewrite sequence  $t_1 \xrightarrow{\mathcal{R}, \mu} s_1 \xrightarrow{*}_{\mathcal{R}, \mu} t_2 \xrightarrow{\mathcal{R}, \mu} \dots$  which contradicts the  $\mu$ -termination of  $\mathcal{R}$ .  $\square$

Proposition 9(3) suggests a simple checking of non- $\mu$ -termination.

**Corollary 4** (Non- $\mu$ -termination criterion). *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . If there is  $u \rightarrow v \in DP_{\mathcal{F}}(\mathcal{R}, \mu)$  such that  $v' = \theta(u)$  for some substitution  $\theta$  and renamed version  $v'$  of  $v$ , then  $\mathcal{R}$  is not  $\mu$ -terminating.*

As a corollary of Theorems 2 and 3, we have:

**Corollary 5** (Characterization of  $\mu$ -termination). *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if there is no infinite minimal  $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$ -chain.*

## 7. Mechanizing proofs of $\mu$ -termination using CSDPs

Over the last 10 years, the dependency pair method has evolved to a powerful technique for proving termination of TRSs in practice. In the DP-approach [10], the starting point is a TRS  $\mathcal{R}$  from which a set of dependency pairs  $DP(\mathcal{R})$  is obtained. Then, these dependency pairs are organized in a dependency graph  $DG(\mathcal{R})$  whose nodes are the pairs in  $DP(\mathcal{R})$  and where the arcs are obtained by investigating possible *rewriting connections* between (instances of) the right-hand sides of the pairs and (instances of) the left-hand sides of other (not necessarily distinct) pairs. The cycles of the graph are analyzed to show that no infinite chains of DPs can be obtained from them [25]. In this sense, the treatment of strongly connected components of the graph (SCCs) instead of cycles [38,39] brought an important improvement to the practical use of this approach.

In the DP-approach, the components  $u_i \rightarrow v_i$  of the chains (or cycles) are dependency pairs, i.e.,  $u_i \rightarrow v_i \in DP(\mathcal{R})$  for all  $i \geq 1$ . Since they only make sense when an underlying TRS  $\mathcal{R}$  is given as the source of the dependency pairs, transforming DPs is possible (the *narrowing* transformation is already described in [10]) but only as a final step because, afterwards, they are no longer dependency pairs of the original TRS. The dependency pair framework [33,35] solves this problem in a clear way, leading to a more powerful mechanization of termination proofs. The central notion now is that of *DP problem* [35, p. 158]: given a TRS  $\mathcal{R}$  and a set of pairs  $\mathcal{P}$ , the goal is to verify the absence of infinite (minimal) chains. In this case, the DP problem is called *finite*. Termination of a TRS  $\mathcal{R}$  is addressed as a DP problem<sup>4</sup>  $(\mathcal{P}, \mathcal{R})$  where  $\mathcal{P} = DP(\mathcal{R})$ :  $\mathcal{R}$  is terminating if this problem is finite. The most important notion regarding the mechanization of the proofs is the notion of *processor*. Formally, a *DP processor* is a function *Proc* that takes a DP problem as input and returns a new set of DP problems that then have to be solved instead. Alternatively, it can also return “no” [35, p. 159]. In the following, we adapt the notions of [35] to CSR.

<sup>4</sup> The original definition in [35] includes an extra parameter  $e$ , which specifies two kinds of problems:  $e = t$  for termination problems, and  $e = i$  for innermost termination problems.

### 7.1. CS problems, CS processors, and the CSDP-framework

In our definition of *DP problem* for *CSR*, we prefer to avoid ‘*DP*’ because, as discussed above, dependency pairs (as such) are relevant in the theoretical framework only for investigating a particular problem (termination of TRSs), whereas some transformations can yield sets of pairs which are no longer dependency pairs of the underlying TRS.

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

**Remark 8.** As in the standard DP framework (see the discussion and further motivation in [35, p. 159]), the inclusion of the case when  $\mathcal{R}$  is nonterminating as part of the definition of infinite problem is essential for dealing with some specific transformations of CS problems (see Theorems 8 and 16).

**Definition 7** (*CS processor*). A CS processor Proc is a mapping from CS problems into sets of CS problems. Alternatively, it can also return “no”. A CS processor Proc is

- *sound* if for all CS problems  $\tau$ , we have that (1)  $\tau$  is finite whenever  $\text{Proc}(\tau) \neq \text{no}$  and (2)  $\forall \tau' \in \text{Proc}(\tau)$ ,  $\tau'$  is finite.
- *complete* if for all CS problems  $\tau$ , we have that (1)  $\tau$  is infinite whenever  $\text{Proc}(\tau) = \text{no}$  or (2)  $\exists \tau' \in \text{Proc}(\tau)$  such that  $\tau'$  is infinite.

A (sound) processor transforms DP problems into (hopefully) *simpler* ones, in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. Here, ‘*simpler*’ usually means that fewer pairs are involved. Soundness is essential for proving *termination*. Completeness is necessary for proving *nontermination*.

Processors are used in a *divide and conquer* scheme to incrementally simplify the original CS problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. The trivial case comes when the set of pairs  $\mathcal{P}$  becomes empty. Then, no infinite chain is possible, and we can provide a *positive* answer yes to the CS problem which is propagated upwards to the original problem in the root of the decision tree. In some cases, it is also possible to witness the existence of infinite chains for a given CS problem; then a *negative* answer no can be provided and propagated upwards.

**Theorem 4** (CSDP-framework). *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . We construct a tree whose nodes are labeled with CS problems or “yes” or “no”, and whose root is labeled with  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\natural})$ . For every inner node labeled with  $\tau$ , there is a sound processor Proc satisfying one of the following conditions:*

1.  $\text{Proc}(\tau) = \text{no}$  and the node has just one child that is labeled with “no”.
2.  $\text{Proc}(\tau) = \emptyset$  and the node has just one child that is labeled with “yes”.
3.  $\text{Proc}(\tau) \neq \text{no}$ ,  $\text{Proc}(\tau) \neq \emptyset$ , and the children of the node are labeled with the CS problems in  $\text{Proc}(\tau)$ .

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

Propositions 8 and 9 are the basis for the following sound and complete processors, which provide some *base cases* for our proofs of termination of *CSR*.

**Theorem 5** (Basic CS processors). *Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (G, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup G}$ . Then, the processors  $\text{Proc}_{\text{Fin}}$  and  $\text{Proc}_{\text{Inf}}$  given by<sup>5</sup>*

$$\text{Proc}_{\text{Fin}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \emptyset & \text{if } P = \emptyset \vee P = \mathcal{P}_X^1 \vee (R = \emptyset \wedge P = \mathcal{P}_X); \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

$$\text{Proc}_{\text{Inf}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \text{no} & \text{if } v = \theta(u) \\ & \text{for some } u \rightarrow v \in \mathcal{P}_G \text{ and substitution } \theta; \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

*are sound and complete.*

<sup>5</sup> In the following, we often write  $\text{Proc}(\mathcal{P}, \mathcal{R}, \mu)$  instead of  $\text{Proc}((\mathcal{P}, \mathcal{R}, \mu))$  to avoid duplicated parentheses.

In the following sections, we describe several sound and (most of them) complete CS processors.

### 8. Context-sensitive dependency graph

In the dependency pairs approach [10], a *dependency graph*  $DG(\mathcal{R})$  is associated to the TRS  $\mathcal{R}$ . The nodes of  $DG(\mathcal{R})$  are the dependency pairs in  $DP(\mathcal{R})$ ; there is an arc from a dependency pair  $u \rightarrow v$  to a dependency pair  $u' \rightarrow v'$  such that  $\text{Var}(u) \cap \text{Var}(u') = \emptyset$  if  $\theta(v) \rightarrow_{\mathcal{R}}^* \theta(u')$  for some substitution  $\theta$ . In [35], a more general notion of *graph of pairs*  $DG(\mathcal{P}, \mathcal{R})$  associated to a set of pairs  $\mathcal{P}$  and a TRS  $\mathcal{R}$  is considered. Pairs in  $\mathcal{P}$  are now used as the nodes of the graph, but they are connected by  $\mathcal{R}$ -rewriting in the same way [35, Definition 7]. The analysis of the cycles in the graph that is built from such pairs is useful for investigating the existence of infinite (minimal) chains of pairs. In the following section, we take into account these points to provide an appropriate definition of context-sensitive (dependency) graph.

#### 8.1. Definition of the context-sensitive dependency graph

Given TRSs  $\mathcal{R}$  and  $\mathcal{P}$  and a replacement map  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ , we want to obtain a notion of graph that is able to represent all infinite *minimal* chains of pairs as given in Definition 5.

**Definition 8** (*Context-sensitive graph of pairs*). Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . The *context-sensitive (CS-)graph*  $G(\mathcal{P}, \mathcal{R}, \mu)$  has  $\mathcal{P}$  as the set of nodes. Given  $u \rightarrow v, u' \rightarrow v' \in \mathcal{P}$ , there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$  if  $u \rightarrow v, u' \rightarrow v'$  is a minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain for some substitution  $\sigma$ .

In termination proofs, we are concerned with the so-called *strongly connected components* (SCCs) of the dependency graph, rather than with the cycles themselves (which are exponentially many) [39]. A strongly connected component in a graph is a *maximal cycle*, i.e., a cycle that is not contained in any other cycle. The following result justifies the use of SCCs for proving the absence of infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chains.

**Theorem 6** (SCC processor). *Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . Then, the processor  $\text{Proc}_{\text{SCC}}$  given by*

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

*is sound and complete.*

**Proof.** We prove soundness by contradiction. Assume that  $\text{Proc}_{\text{SCC}}$  is not sound. Then, there is a CS problem  $\tau = (\mathcal{P}, \mathcal{R}, \mu)$  such that, for all  $\tau' \in \text{Proc}_{\text{SCC}}(\tau)$ ,  $\tau'$  is finite but  $\tau$  is not finite. Thus, there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$ . Since  $\mathcal{P}$  contains a finite number of pairs, there is  $\mathcal{P}' \subseteq \mathcal{P}$  and a tail  $B$  of  $A$ , which is an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu)$ -chain where all pairs in  $\mathcal{P}'$  are infinitely often used. According to Definition 8, this means that  $\mathcal{P}'$  is a cycle in  $G(\mathcal{P}, \mathcal{R}, \mu)$ . Hence  $\mathcal{P}'$  belongs to some SCC with nodes in  $\mathcal{Q}$ , i.e.,  $\mathcal{P}' \subseteq \mathcal{Q}$ . Thus,  $B$  is an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mu)$ -chain, i.e.,  $\tau' = (\mathcal{Q}, \mathcal{R}, \mu)$  is not finite. Since  $\tau' \in \text{Proc}_{\text{SCC}}(\tau)$ , we obtain a contradiction.

With regard to completeness, since  $\mathcal{Q} \subseteq \mathcal{P}$  for some SCC in  $G(\mathcal{P}, \mathcal{R}, \mu)$  with nodes in  $\mathcal{Q}$ , every infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mu)$ -chain is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain. Hence, the processor is complete as well.  $\square$

As a consequence of this theorem, we can *separately* work with the strongly connected components of  $G(\mathcal{P}, \mathcal{R}, \mu)$ , disregarding other parts of the graph. Now we can use these notions to introduce the context-sensitive dependency graph.

**Definition 9** (*Context-sensitive dependency graph*). Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The *Context-Sensitive Dependency Graph* (CSDG) for  $\mathcal{R}$  and  $\mu$  is  $DG(\mathcal{R}, \mu) = G(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^{\sharp})$ .

#### 8.2. Estimating the CS-dependency graph

In general, the context-sensitive graph is *not* computable: it involves reachability of  $\sigma(u')$  from  $\sigma(v)$  (for  $u \rightarrow v \in \mathcal{P}_G$ ) or  $\sigma(t^{\sharp})$  (for  $t \in \mathcal{NHT}_{\mathcal{P}}$ ) using CSR. Since the reachability problem for CSR is undecidable, we need to use some approximation of it.

**Remark 9.** Several estimations of the dependency graph were investigated in [10, 34, 39, 55, 56]. The first one, introduced in [10], was adapted to CSR in [3].

Following [34], we describe how to approximate the CS-dependency graph of a CS-TRS. Given a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , we let  $\text{TCAP}_{\mathcal{R}}^{\mu}$  be as follows:

$$\begin{aligned} \text{TCAP}_{\mathcal{R}}^{\mu}(x) &= y \text{ if } x \text{ is a variable, and} \\ \text{TCAP}_{\mathcal{R}}^{\mu}(f(t_1, \dots, t_k)) &= \begin{cases} f([t_1]_1^f, \dots, [t_k]_k^f) & \text{if } f([t_1]_1^f, \dots, [t_k]_k^f) \text{ does not unify} \\ & \text{with } l \text{ for any } l \rightarrow r \text{ in } \mathcal{R} \\ y & \text{otherwise} \end{cases} \end{aligned}$$

where  $y$  is intended to be a new, fresh variable that has not yet been used and given a term  $s$ ,  $[s]_i^f = \text{TCAP}_{\mathcal{R}}^{\mu}(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ . We assume that  $l$  shares no variable with  $f([t_1]_1^f, \dots, [t_k]_k^f)$  when the unification is attempted. Function  $\text{TCAP}_{\mathcal{R}}^{\mu}$  is intended to provide a suitable approximation of the aforementioned  $(\mathcal{R}, \mu)$ -reachability problems by means of unification. The following result formalizes the correctness of this approach.

**Proposition 10.** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t, u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  be such that  $\text{Var}(t) \cap \text{Var}(u) = \emptyset$ . If  $\theta(t) \hookrightarrow^* \theta(u)$  for some substitution  $\theta$ , then  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$  and  $u$  unify.*

**Proof.** In the following, we let  $s = \text{TCAP}_{\mathcal{R}}^{\mu}(t)$ . Note that, since  $\text{Var}(t) \cap \text{Var}(u) = \emptyset$ , we also have  $\text{Var}(s) \cap \text{Var}(u) = \emptyset$ . Clearly,  $t = \sigma(s)$  for some substitution  $\sigma$ . We proceed by induction on the length  $m$  of the sequence from  $\theta(t)$  to  $\theta(u)$ .

1. If  $m = 0$ , then  $\theta(t) = \theta(\sigma(s)) = \theta(u)$ . Since  $\text{Var}(s) \cap \text{Var}(u) = \emptyset$ , we can write  $\theta(u) = \theta(\sigma(s))$ , i.e.,  $s$  and  $u$  unify.
2. If  $m > 0$ , then we have  $\theta(t) \hookrightarrow t' \hookrightarrow^* \theta(u)$ . Let  $p \in \text{Pos}^{\mu}(\theta(t))$  be the position where the  $\mu$ -rewrite step  $\theta(t) \hookrightarrow t'$  is performed. By definition of  $\text{TCAP}_{\mathcal{R}}^{\mu}$ ,  $s = s[z]_q$  for some fresh variable  $z$  and position  $q$  such that  $q \leq p$ . We can write  $\theta(t) = \theta(s)$ . Furthermore, since  $z$  is a fresh variable, we can write  $t' = \theta(s)$  if we assume that  $\theta(z) = t'[q]$ . Thus,  $\theta(s) \hookrightarrow^* \theta(u)$  in  $m - 1$  steps. By the induction hypothesis,  $\text{TCAP}_{\mathcal{R}}^{\mu}(s)$  and  $u$  unify. Since  $\text{TCAP}_{\mathcal{R}}^{\mu}(s) = \text{TCAP}_{\mathcal{R}}^{\mu}(\text{TCAP}_{\mathcal{R}}^{\mu}(t))$  and  $\text{TCAP}_{\mathcal{R}}^{\mu}(\text{TCAP}_{\mathcal{R}}^{\mu}(t))$  is just a renaming of  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$ , the conclusion follows.  $\square$

According to Proposition 10, given terms  $t, u \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  that share no variable, and a substitution  $\theta$ , the reachability of  $\theta(u)$  from  $\theta(t)$  by  $\mu$ -rewriting can be approximated as unification of  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$  and  $u$ . Thus, taking into account Definitions 5 and 8, we have the following.

**Definition 10** (Estimated context-sensitive graph of pairs). Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . The estimated CS-graph associated to  $\mathcal{R}$  and  $\mathcal{P}$  (denoted  $\text{EG}(\mathcal{P}, \mathcal{R}, \mu)$ ) has  $\mathcal{P}$  as the set of nodes and the arcs that connect them as follows:

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

As a consequence of Proposition 10, we have the following.

**Corollary 6** (Approximation of the context-sensitive graph). Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . The estimated CS-graph  $\text{EG}(\mathcal{P}, \mathcal{R}, \mu)$  contains the CS-graph  $\mathcal{G}(\mathcal{P}, \mathcal{R}, \mu)$ .

Therefore, we have the following estimated CSDG:  $\text{EDG}(\mathcal{R}, \mu) = \text{EG}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^{\sharp})$ .

**Remark 10.** Proposition 10 also provides estimations for  $\mathcal{NHT}_{\mathcal{P}}$ : if  $t \in \mathcal{NHT}_{\mathcal{P}}$ , then  $\text{TCAP}_{\mathcal{R}}^{\mu}(t^{\sharp})$  and  $u$  unify for some  $u \rightarrow v \in \mathcal{P}$ . In the following, we compute  $\mathcal{NHT}_{\mathcal{P}}$  in this way.

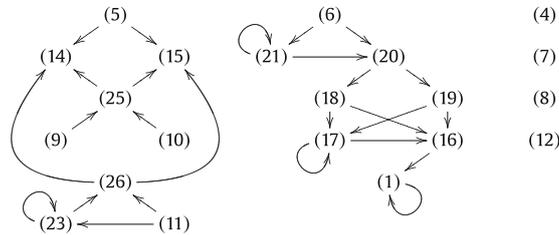


Fig. 5. Context-sensitive dependency graph for the CS-TRS in Example 1.

**Example 15.** Consider again the CS-TRS  $(\mathcal{R}, \mu)$  in Example 1. Note that

$$\mathcal{NHT}_{\text{DP}(\mathcal{R}, \mu)}(\mathcal{R}, \mu^\sharp) = \{\text{oddNs}, \text{incr}(\text{oddNs}), \text{incr}(x), \text{zip}(xs, ys), \text{rep2}(xs)\}$$

The (estimated) CSDG in Fig. 5 has four cycles, each of which contains a single pair. We transform the CS problem  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$  into a set

$$\text{Proc}_{\text{SCC}}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp) = \left\{ \{(1)\}, \mathcal{R}, \mu^\sharp, \{(17)\}, \mathcal{R}, \mu^\sharp, \{(21)\}, \mathcal{R}, \mu^\sharp, \{(23)\}, \mathcal{R}, \mu^\sharp \right\}$$

which contains four new (but very simple) CS problems.

**Remark 11** (CSDG vs. DG). Consider again  $\mathcal{R}$  and  $\mu$  as in Example 1. Pairs (9) and (10) belong to both  $\text{DG}(\mathcal{R})$  (see Fig. 3) and  $\text{DG}(\mathcal{R}, \mu)$  (see Fig. 5). However, they are *not* equally connected in  $\text{DG}(\mathcal{R})$  and  $\text{DG}(\mathcal{R}, \mu)$ . The reason is that the collapsing pair (25), that is *not* a node of  $\text{DG}(\mathcal{R})$ , originates an *incoming* arc from both (9) and (10).

### 9. Treating collapsing pairs

The following result shows how to *safely* transform collapsing pairs into noncollapsing ones in some particular cases.

**Theorem 7** (Removing collapsing pairs). *Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\mathcal{P}' = (\mathcal{G}', P')$  where  $P' = (P - P_X) \cup Q$  for  $Q = \{u \rightarrow t^\sharp \mid u \rightarrow x \in \mathcal{P}_X, t \in \mathcal{NHT}_{\mathcal{P}}\}$ ,  $\mathcal{G}' = \mathcal{G}$  if  $Q = \emptyset$ , and  $\mathcal{G}' = \mathcal{F} \cup \mathcal{G}$  if  $Q \neq \emptyset$ . Then, the processor  $\text{Proc}_{\text{gNHT}}$  given by*

$$\text{Proc}_{\text{gNHT}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}', \mathcal{R}, \mu)\} & \text{if } \mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu) \subseteq \mathcal{T}(\mathcal{F}) \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound.

**Proof.** First, note that  $\mathcal{P}'$  is a TRS: the new rules in  $Q$  are of the form  $u \rightarrow t^\sharp$  for  $t \in \mathcal{NHT}_{\mathcal{P}}$ . Since  $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$ , we trivially have  $\mathcal{V}ar(t^\sharp) \subseteq \mathcal{V}ar(u)$ , i.e.,  $u \rightarrow t^\sharp$  is a rewrite rule. Furthermore, whenever  $Q \neq \emptyset$ ,  $\mathcal{G}'$  is the union of  $\mathcal{F}$  and  $\mathcal{G}$  to reflect the use of symbols in  $\mathcal{F}$  coming from terms  $t^\sharp$  for  $t \in \mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu)$ . Since we assume that  $\mathcal{D}^\sharp \subseteq \mathcal{G}$  (Remark 4),  $\text{root}(t^\sharp) \in \mathcal{D}^\sharp \subseteq \mathcal{G} \subseteq \mathcal{G}'$ .

We prove that the existence of an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain implies the existence of an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu)$ -chain. Consider an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain:

$$\sigma(u_1) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\sharp t_1 \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_2) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\sharp t_2 \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_3) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\sharp \dots$$

for some substitution  $\sigma$ , where, according to Definition 5, for all  $i \geq 1$ ,  $t_i$  is  $\mu$ -terminating and, (1) if  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$  and (2) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{P}_X$ , then  $t_i = s_i^\sharp$  for some  $s_i$  such that  $\sigma(x_i) \triangleright_{\mu} s_i$  and  $s_i = \theta_i(\bar{s}_i)$  for some  $\bar{s}_i \in \mathcal{NHT}$  and substitution  $\theta_i$ . Actually, since  $t_i = s_i^\sharp = \theta_i(\bar{s}_i)^\sharp = \theta_i(\bar{s}_i^\sharp)$  and  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , we can further say that  $\bar{s}_i \in \mathcal{NHT}_{\mathcal{P}}$ .

In case (2), since  $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$ , we have  $t_i = s_i^\sharp = \theta_i(\bar{s}_i^\sharp) = \bar{s}_i^\sharp$ , i.e.,  $t_i \in \mathcal{NHT}_{\mathcal{P}}$ . Thus, we can use  $u_i \rightarrow t_i \in Q$  instead of  $u_i \rightarrow x_i \in \mathcal{P}_X$ , because we still have  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ . In this way, by replacing each  $u_i \rightarrow x_i \in \mathcal{P}_X$  by the corresponding  $u_i \rightarrow t_i \in Q$ , each step  $\sigma(u_i) \xrightarrow{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\sharp t_i$  becomes a step  $\sigma(u_i) \xrightarrow{\mathcal{P}', \mu} t_i$ , whereas steps  $\sigma(u_i) \xrightarrow{\mathcal{P}, \mu} \sigma(v_i) = t_i$  for  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{G}}$  remain unchanged. Thus, we obtain an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu)$ -chain, as desired.  $\square$

Note that no pair in  $\mathcal{P}'$  in Theorem 7 is collapsing. Unfortunately,  $\text{Proc}_{\text{gNHT}}$  is *not* complete.

**Example 16.** Consider the following TRS:

$$\begin{aligned} b &\rightarrow f(c(b)) \\ f(x) &\rightarrow x \end{aligned}$$

together with the replacement map  $\mu$  given by  $\mu(\varepsilon) = \mu(c) = \emptyset$ .  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\begin{aligned} B &\rightarrow F(c(b)) \\ F(x) &\rightarrow x \end{aligned}$$

and  $\mathcal{NHT}_{DP(\mathcal{R}, \mu)} = \{b\}$ . There is no infinite  $(\mathcal{P}, \mathcal{R}, \mu^\sharp)$ -chain for  $\mathcal{P} = DP(\mathcal{R}, \mu)$ , i.e.,  $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$  is finite and  $\mathcal{R}$   $\mu$ -terminating. However, with  $\mathcal{P}'$  as in Theorem 7:

$$\begin{aligned} B &\rightarrow F(c(b)) \\ F(x) &\rightarrow B \end{aligned}$$

we have an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu^\sharp)$ -chain, i.e.,  $(\mathcal{P}', \mathcal{R}, \mu^\sharp)$  is not finite.

The following processor provides a sound and complete transformation of collapsing pairs into noncollapsing pairs.

**Theorem 8** (Transforming collapsing pairs). *Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $u \rightarrow x \in \mathcal{P}_\mathcal{X}$  and*

$$\begin{aligned} P_u &= \{u \rightarrow U(x)\} \\ &\cup \{U(f(x_1, \dots, x_k)) \rightarrow U(x_i) \mid f \in \mathcal{F}, i \in \mu(f)\} \\ &\cup \{U(t) \rightarrow t^\sharp \mid t \in \mathcal{NHT}_\mathcal{P}\} \end{aligned}$$

where  $U$  is a fresh symbol. Let  $\mathcal{P}' = (\mathcal{G} \cup \{U\}, P')$  where  $P' = (P - \{u \rightarrow x\}) \cup P_u$ , and  $\mu'$  which extends  $\mu$  by  $\mu'(U) = \emptyset$ . The processor  $\text{Proc}_{\text{eColl}}$  given by

$$\text{Proc}_{\text{eColl}}(\mathcal{P}, \mathcal{R}, \mu) = \{(\mathcal{P}', \mathcal{R}, \mu')\}$$

is sound and complete.

**Proof.** With regard to soundness, we proceed by contradiction. If  $\text{Proc}_{\text{eColl}}$  is not sound, then there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain but there is no infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain  $A$ . Since  $\mathcal{P}$  is finite, we can assume that there is  $Q \subseteq \mathcal{P}$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \xrightarrow{\Delta}_{Q, \mu} \circ \underset{\mu}{\triangleright} t_1 \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2) \xrightarrow{\Delta}_{Q, \mu} \circ \underset{\mu}{\triangleright} t_2 \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

for some substitution  $\sigma$  and pairs  $u_i \rightarrow v_i \in Q$ , and, for all  $i \geq 1$ ,

1. if  $v_i \notin \mathcal{X}$ , then  $t_i = \sigma(v_i)$ ,
2. if  $v_i = x_i \in \mathcal{X}$ , then  $x_i \notin \text{Var}^\mu(u_i)$ , and  $\sigma(x_i) = C_i[s_i]_{p_i}$  for some context  $C_i[\ ]_{p_i}$ , such that  $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$ ,  $\text{prefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ ,  $s_i = \theta_i(\bar{s}_i)$  for some  $\bar{s}_i \in \mathcal{NHT}_\mathcal{P}$  and substitution  $\theta_i$ , and  $t_i = s_i^\sharp$ .

For 'steps'  $\sigma(u_i) \xrightarrow{\Delta}_{Q, \mu} \circ \underset{\mu}{\triangleright} t_i$  such that  $u_i \rightarrow v_i \neq u \rightarrow x$ , we have  $u_i \rightarrow v_i \in \mathcal{P}'$ . By minimality of  $B$ ,  $t_i$  is  $(\mathcal{R}, \mu)$ -terminating. Since  $t_i \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$  and  $\mu'(f) = \mu(f)$  for all  $f \in \mathcal{F} \cup \mathcal{G}$ ,  $t_i$  is  $(\mathcal{R}, \mu')$ -terminating, too. On the other hand, if  $u_i \rightarrow v_i = u \rightarrow x$ , then, since  $C_i[\ ]_{p_i} \subseteq \mathcal{F}$ ,  $p_i \in \mathcal{P}os^\mu(C_i[\ ]_{p_i})$ , and by definition of  $P_u$ , we get

$$\sigma(u_i) \xrightarrow{*}_{P_u, \mu'} U(\sigma(v_i)) = U(C_i[s_i]_{p_i}) \xrightarrow{*}_{P_u, \mu'} U(s_i) = U(\theta_i(\bar{s}_i)) = \theta_i(U(\bar{s}_i)) \xrightarrow{*}_{P_u, \mu'} \theta_i(\bar{s}_i^\sharp) = s_i^\sharp = t_i$$

where all terms of the form  $U(s)$  in the sequence above are  $(\mathcal{R}, \mu')$ -terminating: since  $\mu'(U) = \emptyset$  and  $U$  does not belong to  $\mathcal{F}$ ,  $U(s)$  is in  $(\mathcal{R}, \mu')$ -normal form. Furthermore, by minimality of  $B$ ,  $t_i$  is  $(\mathcal{R}, \mu)$ -terminating and, since  $\mu'(f) = \mu(f)$  for all  $f \in \mathcal{F} \cup \mathcal{G}$ ,  $t_i$  is  $(\mathcal{R}, \mu')$ -terminating. Therefore, we obtain an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain, leading to a contradiction.

For completeness, we consider two cases: if  $\mathcal{R}$  is not  $\mu$ -terminating, then all termination problems are infinite (both before and after the application of  $\text{Proc}_{\text{eColl}}$ ) and there is no problem. Therefore, assume that  $\mathcal{R}$  is  $\mu$ -terminating and that  $(\mathcal{P}, \mathcal{R}, \mu)$  is finite but there is an infinite  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain. Again, we can assume that there is  $Q \subseteq \mathcal{P}'$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \xrightarrow{\Delta}_{Q, \mu'} \circ \underset{\mu'}{\triangleright} t_1 \xrightarrow{*}_{\mathcal{R}, \mu'} \sigma(u_2) \xrightarrow{\Delta}_{Q, \mu'} \circ \underset{\mu'}{\triangleright} t_2 \xrightarrow{*}_{\mathcal{R}, \mu'} \dots$$

for some substitution  $\sigma$  and pairs  $u_i \rightarrow v_i \in Q$  where  $t_i = \sigma(v_i)$  is  $(\mathcal{R}, \mu')$ -terminating for  $i \geq 1$ . Without loss of generality, we can assume that  $\sigma(x) \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$  for all  $x \in \mathcal{X}$ , i.e.,  $\sigma$  does not introduce any symbol  $U$ . It is not difficult to see that, for each  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain which is based on a substitution  $\sigma'$  whose bindings  $\sigma'(x)$  contain symbols  $U$ , there is a  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain which uses the same pairs in  $\mathcal{P}'$  and rules in  $\mathcal{R}$  for the rewriting steps, but which is based on a substitution  $\sigma$  where the  $U$ 's have been just removed from all bindings  $\sigma'(x)$  to obtain  $\sigma(x)$  instead.

If  $u_i \rightarrow v_i \in P_u$ , then, without loss of generality, we can assume that  $u_i = u$  and  $v_i = U(x)$ . Since  $\mu(U) = \emptyset$ , there is  $n \geq 0$  such that

$$\begin{aligned}
\sigma(u_i) &\xrightarrow{\Delta}_{\mathcal{P}', \mu'} \sigma(U(x)) = U(\sigma(x)) = \sigma(u_{i+1}) \\
&\xrightarrow{\Delta}_{\mathcal{P}', \mu'} \sigma(v_{i+1}) = \sigma(u_{i+2}) \\
&\xrightarrow{\Delta}_{\mathcal{P}', \mu'} \\
&\vdots \\
&\xrightarrow{\Delta}_{\mathcal{P}', \mu'} \sigma(v_{i+n}) = \sigma(U(s_{i+n+1})) \\
&\xrightarrow{\Delta}_{\mathcal{P}', \mu'} \sigma(s_{i+n+1}^\#) = \sigma(t_i) \\
&\xrightarrow{*}_{\mathcal{R}, \mu'} \sigma(u_{i+n+2})
\end{aligned}$$

where, for all  $j, i+1 \leq j \leq i+n$ ,  $u_j = U(f_j(x_1, \dots, x_{i_j}, \dots, x_{k_j}))$ ,  $v_j = U(x_{i_j})$ ,  $i_j \in \mu(f_j)$ , and  $s_{i+n+1} \in \mathcal{NHT}_{\mathcal{P}}$  (by definition of  $\text{Proc}_{\text{eColl}}$ ). Therefore, from the  $n$  rewriting steps that remove the  $f_j \in \mathcal{F}$  for  $1 \leq j \leq n$ , we know that  $\sigma(x) = C_i[t_{i+n+1}]_{p_i}$  with  $p_i \in \text{Pos}^\mu(C_i[\ ]_{p_i})$  and  $\text{sprefix}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$ . Thus, according to Definition 5, we have:  $\sigma(u_i) \xrightarrow{\Delta}_{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\# t_i$  and  $t_i \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_{i+n+2})$ . Furthermore,  $t_i$  is  $\mu$ -terminating (because  $\mathcal{R}$  is  $\mu$ -terminating). On the other hand, if  $u_i \rightarrow v_i \in \mathcal{P} - \{u \rightarrow x\}$ , then we have  $\sigma(u_i) \xrightarrow{\Delta}_{\mathcal{P}, \mu} \circ \triangleright_{\mu}^\# t_i$  satisfying the conditions in Definition 5. We obtain an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain, leading again to a contradiction.  $\square$

**Example 17.** The use of  $\text{Proc}_{\text{eColl}}$  with  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\#)$  in Example 16 yields  $(\mathcal{P}', \mathcal{R}, \mu')$  where  $\mathcal{P}'$  consists of the following pairs:

$$\begin{array}{l}
\text{B} \quad \rightarrow \text{F}(c(b)) \quad \text{F}(x) \rightarrow \text{U}(x) \\
\text{U}(b) \rightarrow \text{B}
\end{array}$$

It is not difficult to see now that there is no infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu')$ -chain.

## 10. Use of $\mu$ -reduction pairs

A reduction pair  $(\succsim, \sqsupset)$  consists of a stable and monotonic quasi-ordering  $\succsim$ , and a stable and well-founded ordering  $\sqsupset$  satisfying either  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$  [42]. The absence of infinite chains of pairs can be ensured by finding a *reduction pair*  $(\succsim, \sqsupset)$  that is compatible with the rules and the pairs:  $l \succsim r$  for all rewrite rules  $l \rightarrow r$  and  $u \succsim v$  or  $u \sqsupset v$  for all dependency pairs  $u \rightarrow v$  [10]. In the dependency pair framework, they are used to obtain *smaller* sets of pairs  $\mathcal{P}' \subseteq \mathcal{P}$  by removing the *strict* pairs, i.e., those pairs  $u \rightarrow v \in \mathcal{P}$  such that  $u \sqsupset v$ .

Stability is required for both  $\succsim$  and  $\sqsupset$  because, although we only check the left- and right-hand sides of the rewrite rules  $l \rightarrow r$  (with  $\succsim$ ) and pairs  $u \rightarrow v$  (with  $\succsim$  or  $\sqsupset$ ), the chains of pairs involve *instances*  $\sigma(l)$ ,  $\sigma(r)$ ,  $\sigma(u)$ , and  $\sigma(v)$  of rules and pairs, and we aim to conclude  $\sigma(l) \succsim \sigma(r)$  and also  $\sigma(u) \succsim \sigma(v)$  or  $\sigma(u) \sqsupset \sigma(v)$ . Monotonicity is required for  $\succsim$  to deal with the application of rules  $l \rightarrow r$  to an arbitrary depth in terms. Since the pairs are 'applied' only at the root level, no monotonicity is required for  $\sqsupset$  (but, for this reason, we cannot compare the rules in  $\mathcal{R}$  using  $\sqsupset$ ). Endrullis et al. noticed that *transitivity* is not necessary for the strict component  $\sqsupset$  because it is somehow 'simulated' by the compatibility requirement above [20].

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

**Definition 11** ( $\mu$ -reduction pair). Let  $\mathcal{F}$  be a signature and  $\mu \in M_{\mathcal{F}}$ . A  $\mu$ -reduction pair  $(\succsim, \sqsupset)$  consists of a stable and  $\mu$ -monotonic quasi-ordering  $\succsim$  and a well-founded stable relation  $\sqsupset$  on terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  that are compatible, i.e.,  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$ . We say that  $(\succsim, \sqsupset)$  is  $\mu$ -monotonic if  $\sqsupset$  is  $\mu$ -monotonic.

The following result allows us to use a  $\mu$ -monotonic  $\mu$ -reduction pair to remove some rewrite rules from the original rewrite system  $\mathcal{R}$  before starting a termination proof.

**Proposition 11** (Removing strict rewrite rules). Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $(\succsim, \sqsupset)$  be a  $\mu$ -monotonic  $\mu$ -reduction pair such that  $l \succsim \cup \sqsupset r$  for all  $l \rightarrow r \in \mathcal{R}$ . Let  $\mathcal{R}_{\sqsupset} = \{l \rightarrow r \in \mathcal{R} \mid l \sqsupset r\}$  and  $S = \mathcal{R} - \mathcal{R}_{\sqsupset}$ . Then,  $\mathcal{R}$  is  $\mu$ -terminating if and only if  $S$  is  $\mu$ -terminating.

**Proof.** Since  $S \subseteq \mathcal{R}$ , the *only if* part is obvious. For the *if* part, we proceed by contradiction. If  $\mathcal{R}$  is not  $\mu$ -terminating, then there is an infinite  $\mu$ -rewrite sequence  $A$ :

$$t_1 \xrightarrow{*}_{\mathcal{R}, \mu} t_2 \xrightarrow{*}_{\mathcal{R}, \mu} \dots t_n \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

where an infinite number of rules in  $\mathcal{R}_{\sqsupset}$  have been used; otherwise, there would be an infinite tail  $t_m \xrightarrow{S, \mu} t_{m+1} \xrightarrow{S, \mu} \dots$  for some  $m \geq 1$  where only rules in  $S$  are applied, contradicting the  $\mu$ -termination of  $S$ . Let  $J = \{j_1, j_2, \dots\}$  be the infinite set of indices indicating  $\mu$ -rewrite steps  $t_j \xrightarrow{\mathcal{R}, \mu} t_{j+1}$  in  $A$ , for all  $j \in J$ , where rules in  $\mathcal{R}_{\sqsupset}$  have been used to perform the  $\mu$ -rewriting step. Since  $l \sqsupset r$  for all  $l \rightarrow r \in \mathcal{R}_{\sqsupset}$ , by stability and  $\mu$ -monotonicity of  $\sqsupset$ , we have that  $t_{j_i} \sqsupset t_{j_{i+1}}$ . Since  $l \succ r$  for all  $l \rightarrow r \in S$ , by stability and  $\mu$ -monotonicity of  $\succ$ , we have that  $t_{j_{i+1}} \succ t_{j_{i+2}}$ . By compatibility between  $\succ$  and  $\sqsupset$ , we have  $t_{j_i} \sqsupset t_{j_{i+1}}$  for all  $i \geq 1$ . We obtain an infinite sequence  $t_{j_1} \sqsupset t_{j_2} \sqsupset \dots$  which contradicts well-foundedness of  $\sqsupset$ .  $\square$

### 10.1. Argument filterings for CSR

An argument filtering  $\pi$  for a signature  $\mathcal{F}$  is a mapping that assigns to every  $k$ -ary function symbol  $f \in \mathcal{F}$  an argument position  $i \in \{1, \dots, k\}$  or a (possibly empty) list  $[i_1, \dots, i_m]$  of argument positions with  $1 \leq i_1 < \dots < i_m \leq k$  [42]. The trivial argument filtering  $\pi_{\top}$  is given by  $\pi_{\top}(f) = [1, \dots, k]$  for each  $k$ -ary symbol  $f \in \mathcal{F}$ . It corresponds to the argument filtering which does nothing. In the dependency pair method, argument filterings  $\pi$  provide a simple way to remove parts of the syntactic structure of a rule  $s \rightarrow t$ . Argument filterings (recursively) drop immediate subterms of terms and can produce terms from a new signature where the arity of symbols has been decreased if necessary. In this way, we obtain simpler expressions that are (hopefully) easy to compare. In the following, we adapt the argument filtering technique to our CSDP framework. In Section 10.2, we investigate their use together with  $\mu$ -reduction pairs. We can use an argument filtering  $\pi$  to ‘filter’ either the signature  $\mathcal{F}$  or any replacement map  $\mu \in M_{\mathcal{F}}$ . In the following, we assume that:

1. The signature  $\mathcal{F}_{\pi}$  consists of all function symbols  $f$  such that  $\pi(f)$  is some list  $[i_1, \dots, i_m]$ , where, in  $\mathcal{F}_{\pi}$ , the arity of  $f$  is  $m$ . As usual, we give the same name to the version of  $f \in \mathcal{F}$  that belongs to  $\mathcal{F}_{\pi}$ .
2. The replacement map  $\mu_{\pi} \in M_{\mathcal{F}_{\pi}}$  is given as follows: for all  $f \in \mathcal{F}$  such that  $f \in \mathcal{F}_{\pi}$  and  $\pi(f) = [i_1, \dots, i_m]$ :

$$\mu_{\pi}(f) = \{j \in \{1, \dots, m\} \mid i_j \in \mu(f)\}$$

An argument filtering  $\pi$  induces a mapping from  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to  $\mathcal{T}(\mathcal{F}_{\pi}, \mathcal{X})$ , also denoted by  $\pi$ :

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

Note that, for the trivial argument filtering  $\pi_{\top}$ , we have that  $\mathcal{F}_{\pi_{\top}} = \mathcal{F}$  and  $\mu_{\pi_{\top}} = \mu$  for all  $\mu \in M_{\mathcal{F}}$ . Furthermore,  $\pi_{\top}(t) = t$  for all  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . In the following, given a substitution  $\sigma$  and an argument filtering  $\pi$ , we let  $\sigma_{\pi}$  be the substitution defined by  $\sigma_{\pi}(x) = \pi(\sigma(x))$  for all  $x \in \mathcal{X}$ . The following auxiliary results are used below.

**Lemma 7.** Let  $\mathcal{F}$  be a signature,  $\pi$  be an argument filtering for  $\mathcal{F}$ , and  $\sigma$  be a substitution. If  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , then  $\pi(\sigma(t)) = \sigma_{\pi}(\pi(t))$ .

**Proof.** By structural induction.

1. Base case:  $t$  is a variable or a constant symbol. If  $t = x \in \mathcal{X}$ , then  $\pi(x) = x$  and  $\pi(\sigma(x)) = \sigma_{\pi}(x) = \sigma_{\pi}(\pi(x))$ . If  $t$  is a constant symbol, then  $\pi(t) = t$  and  $\sigma(t) = t = \sigma_{\pi}(t)$ . Hence,  $\pi(\sigma(t)) = \pi(t) = t = \sigma_{\pi}(t) = \sigma_{\pi}(\pi(t))$ .
2. If  $t = f(t_1, \dots, t_k)$ , then we consider the two possible cases according to  $\pi(f)$ :
  - (a) If  $\pi(f) = i$  for some  $i \in \{1, \dots, k\}$ , then  $\pi(t) = \pi(t_i)$ . By the induction hypothesis,  $\pi(\sigma(t_i)) = \sigma_{\pi}(\pi(t_i))$ . Therefore,  $\pi(\sigma(t)) = \pi(f(\sigma(t_1), \dots, \sigma(t_k))) = \pi(\sigma(t_i)) = \sigma_{\pi}(\pi(t_i)) = \sigma_{\pi}(\pi(f(t_1, \dots, t_k))) = \sigma_{\pi}(\pi(t))$ .
  - (b) If  $\pi(f) = [i_1, \dots, i_m]$ , then  $\pi(t) = f(\pi(t_{i_1}), \dots, \pi(t_{i_m}))$ . By the induction hypothesis,  $\pi(\sigma(t_{i_j})) = \sigma_{\pi}(\pi(t_{i_j}))$  for all  $j \in \{1, \dots, m\}$ . Thus,  $\pi(\sigma(t)) = \pi(f(\sigma(t_1), \dots, \sigma(t_k))) = f(\pi(\sigma(t_{i_1}), \dots, \pi(\sigma(t_{i_m}))) = f(\sigma_{\pi}(\pi(t_{i_1}), \dots, \sigma_{\pi}(\pi(t_{i_m})))) = \sigma_{\pi}(f(\pi(t_{i_1}), \dots, \pi(t_{i_m}))) = \sigma_{\pi}(\pi(t))$ .  $\square$

**Proposition 12.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ ,  $\pi$  be an argument filtering for  $\mathcal{F}$ , and  $s, t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . Let  $\succsim$  be a  $\mu_{\pi}$ -monotonic quasi-ordering such that  $\pi(l) \succsim \pi(r)$  for all  $l \rightarrow r \in R$ . If  $s \xrightarrow{*} t$ , then  $\pi(s) \succsim \pi(t)$ .

**Proof.** By induction on the length  $n$  of the  $\mu$ -rewrite sequence.

1. If  $n = 0$ , then  $s = t$  and, trivially,  $\pi(s) = \pi(t)$ . Since  $\succsim$  is reflexive, we have  $\pi(s) \succsim \pi(t)$ .
2. If  $n > 0$ , we can write  $s \xrightarrow{*} s' \xrightarrow{*} t$ , where the length of the sequence from  $s'$  to  $t$  is  $n - 1$ . Let  $p \in \text{Pos}^{\mu}(s)$  be the  $\mu$ -replacing position where the  $\mu$ -rewriting step  $s \xrightarrow{*} s'$  is performed. We prove that  $s \xrightarrow{*} s'$  implies  $\pi(s) \succsim \pi(s')$  by induction on the structure of  $p$ .

- (a) If  $p = \Lambda$ , then  $s = \sigma(l)$  and  $s' = \sigma(r)$  for some rewrite rule  $l \rightarrow r$  and matching substitution  $\sigma$ . By Lemma 7,  $\pi(s) = \pi(\sigma(l)) = \sigma_\pi(\pi(l))$  and  $\pi(s') = \pi(\sigma(r)) = \sigma_\pi(\pi(r))$ . Since  $\pi(l) \succeq \pi(r)$ , by stability of  $\succeq$  we conclude  $\pi(s) = \sigma_\pi(\pi(l)) \succeq \sigma_\pi(\pi(r)) = \pi(s')$ .
- (b) If  $p = i \cdot q$ , then we can write  $s = f(s_1, \dots, s_i, \dots, s_k)$  and  $s' = f(s'_1, \dots, s'_i, \dots, s'_k)$  for some nonconstant symbol  $f$  (i.e.,  $ar(f) > 0$ ) and we know that  $i \in \mu(f)$ ,  $s_i \hookrightarrow s'_i$  at position  $q$ , and  $s_j = s'_j$  for all  $j \neq i$ . By the induction hypothesis,  $\pi(s_i) \succeq \pi(s'_i)$ . We consider the two possible cases according to  $\pi(f)$ :
- If  $\pi(f) = j$  for some  $j \in \{1, \dots, k\}$ , then  $\pi(s) = \pi(s_j)$ . If  $i \neq j$ , then  $s'_j = s_j$ . By reflexivity of  $\succeq$ , we have  $\pi(s_j) \succeq \pi(s'_j)$ . If  $i = j$ , then we know from above that  $\pi(s_i) \succeq \pi(s'_i)$ . Therefore,  $\pi(s) = \pi(s_j) \succeq \pi(s'_j) = \pi(s')$ .
  - If  $\pi(f) = [i_1, \dots, i_m]$ , then we have that  $\pi(s) = f(\pi(s_{i_1}), \dots, \pi(s_{i_m}))$  and  $\pi(s') = f(\pi(s'_{i_1}), \dots, \pi(s'_{i_m}))$ . Consider  $i_j$  for some  $j \in \{1, \dots, m\}$ . We have two cases:
    - If  $i_j = i$ , then by the induction hypothesis,  $\pi(s_{i_j}) \succeq \pi(s'_{i_j})$  and, by definition of  $\mu_\pi$ ,  $j \in \mu_\pi(f)$ .
    - If  $i_j \neq i$ , then  $s'_{i_j} = s_{i_j}$  and we have  $\pi(s_{i_j}) = \pi(s'_{i_j})$ .
 Note that  $\pi(s_{i_j})$  is the  $j$ th immediate subterm of  $\pi(s)$ . By  $\mu_\pi$ -monotonicity of  $\succeq$ ,

$$\begin{aligned}
 \pi(s) &= \pi(f(s_1, \dots, s_k)) \\
 &= f(\pi(s_{i_1}), \dots, \pi(s_{i_j}), \dots, \pi(s_{i_m})) \\
 &\succeq f(\pi(s_{i_1}), \dots, \pi(s'_{i_j}), \dots, \pi(s_{i_m})) \\
 &= f(\pi(s'_{i_1}), \dots, \pi(s'_{i_j}), \dots, \pi(s'_{i_m})) \\
 &= \pi(f(s'_1, \dots, s'_k)) \\
 &= \pi(s')
 \end{aligned}$$

where we assume that  $i_j = i$  for some  $j \in \{1, \dots, k\}$ . If no such  $j$  exists, then we would have  $\pi(s) = \pi(s')$ , which also implies  $\pi(s) \succeq \pi(s')$  because  $\succeq$  is reflexive.

Thus, we have proved that  $s \hookrightarrow s'$  implies  $\pi(s) \succeq \pi(s')$  as desired.

Therefore,  $\pi(s) \succeq \pi(s')$  and, by the induction hypothesis,  $\pi(s') \succeq \pi(t)$ . By transitivity of  $\succeq$ , we conclude  $\pi(s) \succeq \pi(t)$ .  $\square$

**Remark 12.** We often use argument filterings to transform (sets of) rules  $S$  as follows:  $\pi(s \rightarrow t) = \pi(s) \rightarrow \pi(t)$  for a rule  $s \rightarrow t$ , and  $\pi(S) = \{\pi(s \rightarrow t) \mid s \rightarrow t \in S\}$ . Given a TRS  $\mathcal{R} = (\mathcal{F}, R)$ , we write  $\pi(\mathcal{R})$  to denote the filtered TRS  $(\mathcal{F}_\pi, \pi(R))$ .

### 10.2. Removing pairs using $\mu$ -reduction pairs

Given TRSs  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$ , and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ , checking the absence of infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chains can often be 'simplified' to checking the absence of infinite minimal  $(\mathcal{P}', \mathcal{R}, \mu)$ -chains for a proper subTRS  $\mathcal{P}'$  of  $\mathcal{P}$  by finding appropriate  $\mu$ -reduction pairs  $(\succ, \sqsupset)$ . The presence of collapsing pairs  $u \rightarrow v = u \rightarrow x \in \mathcal{P}_x$  imposes some additional requirements on the  $\mu$ -reduction pairs:

- We need to ensure that the quasi-ordering  $\succ$  is able to 'look' for a  $\mu$ -replacing subterm  $s$  inside the instantiation  $\sigma(x)$  of a migrating variable  $x$ : since  $\sigma(x) = C[s]_p$  for some context  $C[\ ]_p$  and  $\mu$ -replacing position  $p \in \mathcal{P}os^\mu(C[\ ]_p)$  such that  $\text{spreff}_{C[\ ]_p}(p) \subseteq \mathcal{F}$ , we can obtain  $s$  out from  $C[s]_p$  by applying the projection rules in  $\text{Emb}^\mu(\mathcal{F})$  (Definition 1). Hence, we require  $\text{Emb}^\mu(\mathcal{F}) \subseteq \succ$ .
- We need to connect the marked version  $s^\sharp$  of  $s$  (which is an instance of a hidden term  $t \in \mathcal{NHT}_\mathcal{P}$ , i.e.,  $s = \theta(t)$  for some substitution  $\theta$ ) with an instance  $\sigma(u)$  of the left-hand side  $u$  of a pair; hence, we require  $t \succeq t^\sharp$  or  $t \sqsupset t^\sharp$  for all  $t \in \mathcal{NHT}_\mathcal{P}$  which, by stability, becomes  $s \succeq s^\sharp$  or  $s \sqsupset s^\sharp$ .

The following theorem formalizes a generic processor to remove pairs from  $\mathcal{P}$  by using argument filterings and  $\mu$ -reduction pairs.

**Theorem 9** ( $\mu$ -reduction pair processor). *Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\pi$  be an argument filtering for  $\mathcal{F} \cup \mathcal{G}$  and  $(\succ, \sqsupset)$  be a  $\mu_\pi$ -reduction pair such that*

- $\pi(\mathcal{R}) \subseteq \succ, \pi(\mathcal{P}) \subseteq \succ \cup \sqsupset$ , and
- whenever  $\mathcal{NHT}_\mathcal{P} \neq \emptyset$  and  $\mathcal{P}_x \neq \emptyset$ , we have that
  - for all  $f \in \mathcal{F}$ , either  $\pi(f) = [i_1, \dots, i_m]$  and  $\mu(f) \subseteq \pi(f)$ , or  $\pi(f) = i$  and  $\mu(f) = \{i\}$ ,
  - $\text{Emb}^{\mu_\pi}(\mathcal{F}_\pi) \subseteq \succ$ , and

(c)  $\pi(t) (\succcurlyeq \cup \sqsupset) \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ ,

Let  $\mathcal{P}_{\sqsupset} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \sqsupset \pi(v)\}$ . Then, the processor  $\text{Proc}_{\text{RP}}$  given by

$$\text{Proc}_{\text{RP}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_{\sqsupset}, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** Completeness is obvious, since  $\mathcal{P} - \mathcal{P}_{\sqsupset} \subseteq \mathcal{P}$ . Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$ , but that there is no infinite minimal  $(\mathcal{P} - \mathcal{P}_{\sqsupset}, \mathcal{R}, \mu)$ -chain. Due to the finiteness of  $\mathcal{P}$ , we assume that there is  $\mathcal{Q} \subseteq \mathcal{P}$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \xrightarrow{\mathcal{Q}, \mu} \circ \overset{\sharp}{\succcurlyeq} t_1 \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_2) \xrightarrow{\mathcal{Q}, \mu} \circ \overset{\sharp}{\succcurlyeq} t_2 \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_3) \xrightarrow{\mathcal{Q}, \mu} \circ \overset{\sharp}{\succcurlyeq} \dots$$

for some substitution  $\sigma$ , where all pairs in  $\mathcal{Q}$  are infinitely often used. Also, for all  $i \geq 1$ , (1) if  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$  and (2) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$ , then  $t_i = s_i^\sharp$  for some  $s_i$  such that  $\sigma(x_i) = C_i[s_i]_{p_i}$  for some  $C_i[\ ]_{p_i}$  and  $p_i \in \text{Pos}^\mu(C_i[\ ]_{p_i})$  such that  $\text{spre}f_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$  and  $s_i = \theta_i(\bar{s}_i)$  for some  $\bar{s}_i \in \mathcal{NHT}$  and substitution  $\theta_i$ . Actually, since  $t_i = s_i^\sharp = \theta_i(\bar{s}_i)^\sharp = \theta_i(\bar{s}_i^\sharp)$  and  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , we can further say that  $\bar{s}_i \in \mathcal{NHT}_{\mathcal{Q}}$ .

Since  $\pi(u_i) (\succcurlyeq \cup \sqsupset) \pi(v_i)$  for all  $u_i \rightarrow v_i \in \mathcal{Q} \subseteq \mathcal{P}$ , by stability of  $\succcurlyeq$  and  $\sqsupset$ , we have  $\sigma_\pi(\pi(u_i)) (\succcurlyeq \cup \sqsupset) \sigma_\pi(\pi(v_i))$  for all  $i \geq 1$ .

No pair  $u \rightarrow v \in \mathcal{Q}$  satisfies that  $\pi(u) \sqsupset \pi(v)$ . Otherwise, we get a contradiction by considering the following two cases:

1. If  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i) \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$  and by Proposition 12,  $\pi(t_i) \succcurlyeq \pi(\sigma(u_{i+1}))$ . By Lemma 7,  $\pi(t_i) \succcurlyeq \sigma_\pi(\pi(u_{i+1}))$ . Since we have  $\sigma_\pi(\pi(u_i)) (\succcurlyeq \cup \sqsupset) \sigma_\pi(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$  (using Lemma 7), by using transitivity of  $\succcurlyeq$  and compatibility between  $\succcurlyeq$  and  $\sqsupset$ , we conclude that  $\sigma_\pi(\pi(u_i)) (\succcurlyeq \cup \sqsupset) \sigma_\pi(\pi(u_{i+1}))$ .
2. If  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$ , then  $\sigma(v_i) = \sigma(x_i) = C_i[s_i]_{p_i}$ . Since  $i \in \mu(f)$  implies that  $i \in \pi(f)$ , we can say that  $\pi(\sigma(x_i)) = \sigma_\pi(x_i) = \pi(C_i[\ ]_{p_i})_{q_i}$  for some  $q_i \in \text{Pos}^{\mu_\pi}(\pi(C_i))$  and  $\text{spre}f_{\pi(C_i)}(q_i) \subseteq \mathcal{F}_\pi$ . Since  $\varepsilon \text{mb}^{\mu_\pi}(\mathcal{F}_\pi) \subseteq \succcurlyeq$ , we have  $\sigma_\pi(\pi(v_i)) = \sigma_\pi(x_i) \succcurlyeq \pi(s_i)$ . Furthermore, we are assuming that  $\pi(t) (\succcurlyeq \cup \sqsupset) \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_{\mathcal{Q}} \subseteq \mathcal{NHT}_{\mathcal{P}}$ . Since  $s_i = \theta_i(\bar{s}_i)$ , we have that  $\pi(s_i) = \pi(\theta_i(\bar{s}_i)) = \theta_{i,\pi}(\pi(\bar{s}_i))$  (using Lemma 7 again) and, similarly,  $\pi(s_i^\sharp) = \theta_{i,\pi}(\pi(\bar{s}_i^\sharp))$ . By stability we have that  $\pi(s_i) (\succcurlyeq \cup \sqsupset) \pi(s_i^\sharp)$ . Hence, by transitivity of  $\succcurlyeq$  (and compatibility of  $\succcurlyeq$  and  $\sqsupset$ ), we have  $\sigma_\pi(\pi(v_i)) = \sigma_\pi(x_i) (\succcurlyeq \cup \sqsupset) \pi(s_i^\sharp)$ . Finally, since  $\pi(s_i^\sharp) = \pi(t_i)$  and  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$  for all  $i \geq 1$ , by Proposition 12 and Lemma 7,  $\pi(t_i) \succcurlyeq \sigma_\pi(\pi(u_{i+1}))$ . Therefore, again by transitivity of  $\succcurlyeq$  and compatibility of  $\succcurlyeq$  and  $\sqsupset$ , we conclude that  $\sigma_\pi(\pi(u_i)) (\succcurlyeq \cup \sqsupset) \sigma_\pi(\pi(u_{i+1}))$ .

Since  $u \rightarrow v$  occurs infinitely often in  $B$ , there is an infinite set  $\mathcal{I} \subseteq \mathbb{N}$  such that  $\sigma_\pi(\pi(u_i)) \sqsupset \sigma_\pi(\pi(u_{i+1}))$  for all  $i \in \mathcal{I}$ . And we have  $\sigma_\pi(\pi(u_i)) (\succcurlyeq \cup \sqsupset) \sigma_\pi(\pi(u_{i+1}))$  for all other  $u_i \rightarrow v_i \in \mathcal{Q}$ . Thus, by using the compatibility conditions of the  $\mu_\pi$ -reduction pair, we obtain an infinite decreasing  $\sqsupset$ -sequence that contradicts the well-foundedness of  $\sqsupset$ .

Therefore,  $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_{\sqsupset}$ , which means that  $B$  is an infinite minimal  $(\mathcal{P} - \mathcal{P}_{\sqsupset}, \mathcal{R}, \mu)$ -chain, thus leading to a contradiction.  $\square$

**Example 18.** Consider the TRS  $\mathcal{R}$  [63, Example 5]:

$$\begin{aligned} \text{if}(\text{true}, x, y) &\rightarrow x & \text{f}(x) &\rightarrow \text{if}(x, c, \text{f}(\text{true})) \\ \text{if}(\text{false}, x, y) &\rightarrow y \end{aligned}$$

with  $\mu(\text{f}) = \{1\}$  and  $\mu(\text{if}) = \{1, 2\}$ . Then,  $\text{DP}(\mathcal{R}, \mu)$  consists of the following CSDPs:

$$\text{F}(x) \rightarrow \text{IF}(x, c, \text{f}(\text{true})) \quad \text{IF}(\text{false}, x, y) \rightarrow y$$

with  $\mu^\sharp(\text{F}) = \{1\}$  and  $\mu^\sharp(\text{IF}) = \{1, 2\}$ . The  $\mu$ -reduction pair  $(\succcurlyeq, >)$  induced by the polynomial interpretation<sup>6</sup>

$$\begin{aligned} [c] &= [\text{true}] = 0 & [\text{f}](x) &= x & [\text{F}](x) &= x \\ [\text{false}] &= 1 & [\text{if}](x, y, z) &= x + y + z & [\text{IF}](x, y, z) &= x + z \end{aligned}$$

<sup>6</sup> See [49] for more information about the automatic generation of polynomial (quasi-)orderings with monotonicity requirements specified by means of replacement maps.

can be used to prove the  $\mu$ -termination of  $\mathcal{R}$ . For  $\mathcal{P} = \text{DP}(\mathcal{R}, \mu)$ , we have  $\mathcal{NHT}_{\mathcal{P}} = \{\varepsilon(\text{true})\}$ . First, we can see that the quasi-ordering is compatible with the rules in  $\text{Emb}^{\mu}(\mathcal{F})$ :

$$\begin{aligned} [\varepsilon(x)] &= x && \geq x = [x] \\ [\text{if}(x, y, z)] &= x + y + z && \geq x = [x] \\ [\text{if}(x, y, z)] &= x + y + z && \geq y = [y] \end{aligned}$$

Now we can see that the condition on the only hidden term in  $\mathcal{NHT}_{\mathcal{P}}$  is also fulfilled:

$$[\varepsilon(\text{true})] = 0 \geq 0 = [\varepsilon(\text{true})]$$

Finally, for the three rules in  $\mathcal{R}$  and the two pairs in  $\mathcal{P}$ , we have:

$$\begin{aligned} [\varepsilon(x)] &= x && \geq x = [\text{if}(x, c, \varepsilon(\text{true}))] \\ [\text{if}(\text{true}, x, y)] &= x + y && \geq x = [x] \\ [\text{if}(\text{false}, x, y)] &= x + y + 1 && \geq y = [y] \\ [\varepsilon(x)] &= x && \geq x = [\text{IF}(x, c, \varepsilon(\text{true}))] \\ [\text{IF}(\text{false}, x, y)] &= y + 1 && > y = [y] \end{aligned}$$

We remove the 'strict' pair  $\text{IF}(\text{false}, x, y) \rightarrow y$  from  $\mathcal{P}$  to obtain  $\mathcal{P}'$ . With  $(\mathcal{P}', \mathcal{R}, \mu^{\sharp})$ , the application of  $\text{Proc}_{\text{SCC}}$  leads to an empty set of CS problems. Thus, the  $\mu$ -termination of  $\mathcal{R}$  is proved.

The 'compatibility' between the replacement map  $\mu$  and the argument filtering  $\pi$ , which is required when collapsing pairs are present, is necessary in Theorem 9.

**Example 19.** Consider the following TRS  $\mathcal{R}$ :

$$\begin{aligned} a &\rightarrow c(\text{h}(\varepsilon(a), b)) \\ \varepsilon(c(x)) &\rightarrow x \end{aligned}$$

together with the replacement map  $\mu$  given by  $\mu(\varepsilon) = \mu(\text{h}) = \{1\}$  and  $\mu(c) = \emptyset$ .  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\varepsilon(c(x)) \rightarrow x$$

and  $\mathcal{NHT}_{\text{DP}(\mathcal{R}, \mu)} = \{\varepsilon(a)\}$ . Note that  $\mathcal{R}$  is *not*  $\mu$ -terminating:

$$\varepsilon(\underline{a}) \hookrightarrow \varepsilon(c(\text{h}(\varepsilon(\underline{a}), b))) \hookrightarrow \text{h}(\varepsilon(\underline{a}), b) \hookrightarrow \dots$$

For the argument filtering  $\pi$  given by  $\pi(a) = \pi(\text{h}) = []$ ,  $\pi(\varepsilon) = \pi(\varepsilon) = [1]$  and  $\pi(c) = 1$ ,  $\mathcal{F}_{\pi}$  consists of the constants  $a$ ,  $\text{h}$  and symbol  $\varepsilon$  of arity 1. Also,  $\mu_{\pi}^{\sharp}(\varepsilon) = \mu_{\pi}^{\sharp}(\varepsilon) = \{1\}$  and  $\mu_{\pi}^{\sharp}(a) = \mu_{\pi}^{\sharp}(\text{h}) = \emptyset$ . We get the constraints:

$$\begin{aligned} \pi(a) &= a && \succ \text{h} &= \pi(c(\text{h}(\varepsilon(a), b))) \\ \pi(\varepsilon(c(x))) &= \varepsilon(x) && \succ x &= \pi(x) \\ &&& \varepsilon(x) &\succ x \\ \pi(\varepsilon(a)) &= \varepsilon(a) && \succ \varepsilon(a) &= \pi(\varepsilon(a)) \\ \pi(\varepsilon(c(x))) &= \varepsilon(x) && \sqsupset x &= \pi(x) \end{aligned}$$

which are easily satisfiable (by a polynomial interpretation, for instance). We would wrongly conclude  $\mu$ -termination of  $\mathcal{R}$ . Note that  $\pi(c) = 1$  but  $\mu^{\sharp}(c) = \emptyset$ , and that  $\pi(\text{h}) = []$  but  $\mu^{\sharp}(\text{h}) = \{1\}$ .

The next processor is useful when all (filterings of) terms in  $\mathcal{NHT}_{\mathcal{P}}$  are *ground*. The advantage is that the quasi-ordering  $\succ$  of the  $\mu$ -reduction pair does not need to impose compatibility with the rules in  $\text{Emb}^{\mu}(\mathcal{F})$ .

**Theorem 10** ( $\mu$ -reduction pair processor for ground hidden terms). *Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\pi$  be an argument filtering for  $\mathcal{F} \cup \mathcal{G}$  such that, for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ ,  $\pi(t)$  is ground. Let  $(\succ, \sqsupset)$  be a  $\mu_{\pi}$ -reduction pair such that*

1.  $\pi(\mathcal{R}) \subseteq \succsim, \pi(\mathcal{P}_G) \subseteq \succsim \cup \sqsupset$ , and
2. for all  $u \rightarrow v \in \mathcal{P}_X$  and all  $t \in \mathcal{NHT}_P$ ,  $\pi(u) (\succsim \cup \sqsupset) \pi(t^\sharp)$

Let  $\mathcal{P}_\sqsupset = \{u \rightarrow v \in \mathcal{P}_G \mid \pi(u) \sqsupset \pi(v)\} \cup \{u \rightarrow v \in \mathcal{P}_X \mid \forall t \in \mathcal{NHT}_P, \pi(u) \sqsupset \pi(t^\sharp)\}$ . Then, the processor  $\text{Proc}_{\text{CRPG}}$  given by

$$\text{Proc}_{\text{CRPG}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_\sqsupset, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** The proof is analogous to that of Theorem 9. Assume the facts and notation in the first paragraph of such a proof. Again, we proceed by contradiction and assume that a pair  $u \rightarrow v \in \mathcal{Q}$  is in  $\mathcal{P}_\sqsupset$ . Again, we have  $\sigma_\pi(\pi(u_i)) (\succsim \cup \sqsupset) \sigma_\pi(\pi(u_{i+1}))$  for all pairs  $u_i \rightarrow v_i \in \mathcal{Q}_G$ .

Now, if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_X$ , then since  $\pi(u_i) (\succsim \cup \sqsupset) \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_Q \subseteq \mathcal{NHT}_P$ , by stability we have that  $\sigma_\pi(\pi(u_i)) (\succsim \cup \sqsupset) \sigma_\pi(\pi(t^\sharp))$ . Since  $\pi(t)$  is ground, we have  $\sigma_\pi(\pi(u_i)) (\succsim \cup \sqsupset) \pi(t^\sharp)$ . Therefore, since  $s_i \in \mathcal{NHT}_Q$  and  $t_i = s_i^\sharp$ , we have  $\sigma_\pi(\pi(u_i)) (\succsim \cup \sqsupset) \pi(t_i)$ . Finally, since  $s_i^\sharp = t_i$  and  $t_i \xrightarrow[\mathcal{R}, \mu]{*} \sigma(u_{i+1})$  for all  $i \geq 1$ , by Proposition 12 and Lemma 7, we have that  $\pi(t_i) \succsim \sigma_\pi(\pi(u_{i+1}))$ . Thus, we also have  $\sigma_\pi(\pi(u_i)) (\succsim \cup \sqsupset) \sigma_\pi(\pi(u_{i+1}))$ .

Since  $u \rightarrow v$  occurs infinitely often in  $B$ , by using the compatibility conditions of the  $\mu_\pi$ -reduction pair, we obtain an infinite decreasing  $\sqsupset$ -sequence that contradicts well-foundedness of  $\sqsupset$ . In particular, if  $u \rightarrow v \in \mathcal{Q}_X \cap \mathcal{P}_\sqsupset$ , then  $\pi(u) \sqsupset \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_Q$ , so each time that  $u \rightarrow v$  is used, a strict decrease occurs.  $\square$

Theorem 10 can succeed when Theorem 9 fails.

**Example 20.** Consider the TRS  $\mathcal{R}$ :

$$a \rightarrow f(d(c(a))) \tag{27}$$

$$f(c(x)) \rightarrow x \tag{28}$$

$$d(c(x)) \rightarrow b \tag{29}$$

and the replacement map  $\mu$  given by  $\mu(c) = \emptyset$  and  $\mu(f) = \mu(d) = \{1\}$ . There are three CSDPs:

$$A \rightarrow F(d(c(a))) \tag{30}$$

$$A \rightarrow D(c(a)) \tag{31}$$

$$F(c(x)) \rightarrow x \tag{32}$$

$\text{Proc}_{\text{SCC}}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\sharp)$  yields a single CS problem  $(\mathcal{P}, \mathcal{R}, \mu)$  with  $\mathcal{P} = \{(30), (32)\}$ . Since  $\mathcal{NHT}_P = \{a\} \neq \emptyset$  and  $F(c(x)) \rightarrow x$  is a collapsing CSDP, according to Theorem 9 we would require that any  $\mu$ -reduction ordering used in the theorem satisfy  $\mathcal{E}mb^\mu(\mathcal{F}) \subseteq \succsim$  (assume the trivial filtering  $\pi_\top$  here) and that  $a (\succsim \cup \sqsupset) A$ . In this case, though, since  $d(c(a)) \succeq_\mu c(a)$ , we must have  $d(c(a)) \succsim c(a)$ ; by  $\mu$ -monotonicity of  $\succsim$ ,  $F(d(c(a))) \succsim F(c(a))$ . Now, one of the following two cases must hold:

1.  $A \sqsupset F(d(c(a)))$  and  $F(c(x)) (\succsim \cup \sqsupset) x$ . By stability of  $\succsim$  and  $\sqsupset$ , we have  $F(c(a)) (\succsim \cup \sqsupset) a$ . Thus,

$$A \sqsupset F(d(c(a))) \succsim F(c(a)) (\succsim \cup \sqsupset) a (\succsim \cup \sqsupset) A.$$

- By compatibility of  $\succsim$  and  $\sqsupset$ , we have  $A \sqsupset \dots \sqsupset A$ , contradicting the well-foundedness of  $\sqsupset$ .
2.  $A (\succsim \cup \sqsupset) F(d(c(a)))$  and  $F(c(x)) \sqsupset x$ . Hence,

$$A (\succsim \cup \sqsupset) F(d(c(a))) \succsim F(c(a)) \sqsupset a (\succsim \cup \sqsupset) A.$$

Again, by compatibility of  $\succsim$  and  $\sqsupset$ , we have  $A \sqsupset \dots \sqsupset A$ .

Thus, Theorem 9 cannot be used with this example. Since  $\mathcal{NHT}_P \subseteq \mathcal{T}(\mathcal{F})$ , Theorem 10 is applicable here. The  $\mu$ -reduction pair  $(\succsim, >)$  induced by the following polynomial interpretation:<sup>7</sup>

<sup>7</sup> See [49,51] for details about the use of polynomial interpretations with rational coefficients.

$$\begin{array}{llll} [\mathbf{a}] & = 1 & [\mathbf{b}] & = 0 & [\mathbf{c}](x) & = x + 1 & [\mathbf{d}](x) & = \frac{1}{4}x \\ [\mathbf{f}](x) & = x & [\mathbf{A}] & = 1 & [\mathbf{F}](x) & = 0 \end{array}$$

can be used to remove (30) from  $\mathcal{P}$ . For the three rules in  $\mathcal{R}$  and the two pairs in  $\mathcal{P}$ , we have:

$$\begin{array}{llll} [\mathbf{a}] & = 1 & \geq \frac{1}{2} & = [\mathbf{f}(\mathbf{d}(\mathbf{c}(\mathbf{a})))] \\ [\mathbf{f}(\mathbf{c}(x))] & = x + 1 & \geq x & = [\mathbf{x}] \\ [\mathbf{d}(\mathbf{c}(x))] & = \frac{1}{4}x + \frac{1}{4} & \geq 0 & = [\mathbf{b}] \\ [\mathbf{A}] & = 1 & > \frac{1}{2} & = [\mathbf{F}(\mathbf{d}(\mathbf{c}(\mathbf{a})))] \\ [\mathbf{F}(\mathbf{c}(x))] & = x + 1 & \geq 1 & = [\mathbf{A}] \end{array}$$

We remove (30) from  $\mathcal{P}$  to obtain  $\mathcal{P}' = \{(32)\}$ . Now,  $\text{Proc}_{\text{SCC}}(\mathcal{P}', \mathcal{R}, \mu) = \emptyset$  because  $\mathcal{NHT}_{\mathcal{P}'} = \emptyset$  and  $\text{EG}(\mathcal{P}', \mathcal{R}, \mu)$  contains no cycle. Thus, the  $\mu$ -termination of  $\mathcal{R}$  is proved.

Nevertheless, even with  $\mathcal{NHT}_{\mathcal{P}} \subseteq \mathcal{T}(\mathcal{F})$ , Theorem 9 can be helpful when Theorem 10 fails.

**Example 21.** Consider  $\mathcal{R}$  and  $\mu$  as in Example 16. Theorem 10 cannot be used here because, reasoning as in Example 16, we would obtain constraints that are incompatible with the well-foundedness of  $\sqsupset$  for any strict component  $\sqsupset$  of a  $\mu$ -reduction pair  $(\succ, \sqsupset)$ . However, the  $\mu$ -termination of  $\mathcal{R}$  can be easily proved with Theorem 9. The  $\mu$ -reduction pair  $(\succ, >)$  generated by the following polynomial interpretation:

$$\begin{array}{lll} [\mathbf{b}] & = 1 & [\mathbf{c}](x) & = 0 & [\mathbf{f}](x) & = x \\ [\mathbf{B}] & = 2 & [\mathbf{F}](x) & = x + 1 \end{array}$$

satisfies the requirements of Theorem 10 and can be used to show a weak decrease of the rules and a strict decrease of the two CSDPs which can both be removed.

Our last result establishes that if we are able to provide a strict comparison between unmarked and marked versions of the (filtered) hidden terms in  $\mathcal{NHT}_{\mathcal{P}}$ , then we can remove *all* collapsing pairs at the same time.

**Theorem 11** ( $\mu$ -reduction pair processor for collapsing pairs). *Let  $\mathcal{R} = (\mathcal{F}, \mathcal{R})$  and  $\mathcal{P} = (\mathcal{G}, \mathcal{P})$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\pi$  be an argument filtering for  $\mathcal{F} \cup \mathcal{G}$  and  $(\succ, \sqsupset)$  be a  $\mu_{\pi}$ -reduction pair such that*

1.  $\pi(\mathcal{R}) \subseteq \succ, \pi(\mathcal{P}) \subseteq \succ \cup \sqsupset$ ,
2.  $\pi(t) \sqsupset \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$  and
  - (a) for all  $f \in \mathcal{F}$ , either  $\pi(f) = [i_1, \dots, i_m]$  and  $\mu(f) \subseteq \pi(f)$ , or  $\pi(f) = i$  and  $\mu(f) = \{i\}$ ,
  - (b)  $\text{Emb}^{\mu_{\pi}}(\mathcal{F}_{\pi}) \subseteq \succ$ .

Then, the processor  $\text{Proc}_{\text{RPC}}$  given by

$$\text{Proc}_{\text{RPC}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** As in the proof of Theorem 9, we proceed by contradiction. We assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$ , but that there is no infinite minimal  $(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)$ -chain. Thus, there is  $\mathcal{Q} \subseteq \mathcal{P}$  such that  $\mathcal{Q} \cap \mathcal{P}_{\mathcal{X}} \neq \emptyset$  and  $A$  has a tail  $B$  as in the proof of Theorem 9. Now, we assume the notation as in the first paragraph of such a proof.

We have  $\sigma_{\pi}(\pi(u_i)) (\succ \cup \sqsupset) \pi(t_i)$  and  $\pi(t_i) \succ \sigma_{\pi}(\pi(u_{i+1}))$  for all pairs  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{G}}$ . If  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$ , then by applying the considerations in the corresponding item of the proof of Theorem 9 and taking into account that  $\pi(t) \sqsupset \pi(t^\sharp)$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ , we now have that  $\sigma_{\pi}(\pi(u_i)) (\succ \cup \sqsupset) \sigma_{\pi}(x_i) \sqsupset \pi(t_i) \succ \sigma_{\pi}(\pi(u_{i+1}))$ . Since pairs  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{X}}$  occur infinitely often in  $B$ , by using the compatibility conditions of the  $\mu_{\pi}$ -reduction pair, we obtain an infinite decreasing  $\sqsupset$ -sequence that contradicts the well-foundedness of  $\sqsupset$ .  $\square$

### 11. Subterm criterion

In [38], Hirokawa and Middeldorp introduce a *subterm criterion* that permits certain cycles of the dependency graph to be ignored *without paying attention to the rules of the TRS*. Their result applies to *cycles* in the *dependency graph*. Thiemann has adapted it to the DP-framework [62, Section 4.6]. In our adaptation to CSR, we take ideas from both works. Our first definition is inspired by Thiemann's *head symbols* [62, Definition 4.36].

**Definition 12** (*Root symbols of a TRS*). Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS. The set of *root symbols* associated to  $\mathcal{R}$  is:

$$\text{Root}(\mathcal{R}) = \{\text{root}(l) \mid l \rightarrow r \in R\} \cup \{\text{root}(r) \mid l \rightarrow r \in R, r \notin \mathcal{X}\}$$

The following result relates  $\text{Root}(\mathcal{P})$  and the set  $\mathcal{H}_{\mathcal{P}}$  of hidden symbols occurring at the root of terms in  $\mathcal{NHT}_{\mathcal{P}}(\mathcal{R}, \mu)$ . It is silently used in the statements of some theorems below.

**Lemma 8.** Let  $\mathcal{R} = (\mathcal{F}, R) = (C \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (G, P)$  be TRSs such that  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . For all  $f \in \mathcal{H}_{\mathcal{P}}$ , we have  $f^{\sharp} \in \text{Root}(\mathcal{P})$ .

**Proof.** If  $f \in \mathcal{H}_{\mathcal{P}}$ , then there is  $t \in \mathcal{NHT}_{\mathcal{P}}$  such that  $f = \text{root}(t)$ . Therefore, there are substitutions  $\theta$  and  $\theta'$  such that  $\theta(t^{\sharp}) \xrightarrow{*}_{\mathcal{R}, \mu} \theta'(u)$  for some  $u \rightarrow v \in \mathcal{P}$ . Since  $f^{\sharp} \notin \mathcal{F}$ ,  $\mu$ -rewritings on  $\theta(t^{\sharp})$  using  $\mathcal{R}$  do not remove it. Thus,  $\text{root}(u) = f^{\sharp}$  and  $f^{\sharp} \in \text{Root}(\mathcal{P})$ .  $\square$

Thiemann uses argument filterings (see Section 10.1) instead of *simple projections* [38, Definition 10]. We find it more convenient to follow Hirokawa and Middeldorp's style, so we generalize their definition to be used with TRSs rather than cycles in the dependency graph.

**Definition 13** (*Simple projection*). Let  $\mathcal{R}$  be a TRS. A *simple projection* for  $\mathcal{R}$  is a mapping  $\pi$  that assigns to every  $k$ -ary symbol  $f \in \text{Root}(\mathcal{R})$  an argument position  $i \in \{1, \dots, k\}$ . The mapping that assigns a subterm  $\pi(t) = t|_{\pi(f)}$  to every term  $t = f(t_1, \dots, t_k)$  with  $f \in \text{Root}(\mathcal{R})$  is also denoted by  $\pi$ ; we also let  $\pi(x) = x$  if  $x \in \mathcal{X}$ .

Given a simple projection  $\pi$  for a TRS  $\mathcal{R}$ , we let  $\pi(\mathcal{R}) = \{\pi(l) \rightarrow \pi(r) \mid l \rightarrow r \in \mathcal{R}\}$ .

**Theorem 12** (*Subterm processor for noncollapsing pairs*). Let  $\mathcal{R} = (\mathcal{F}, R) = (C \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (G, P)$  be TRSs such that  $\mathcal{P}$  contains no collapsing rule, i.e., for all  $u \rightarrow v \in \mathcal{P}$ ,  $v \notin \mathcal{X}$ , and  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ . Let  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$  and let  $\pi$  be a simple projection for  $\mathcal{R}$ . Let  $\mathcal{P}_{\pi, \triangleright_{\mu}} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright_{\mu} \pi(v)\}$ . Then, the processor  $\text{Proc}_{\text{SubNColl}}$  given by

$$\text{Proc}_{\text{SubNColl}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_{\pi, \triangleright_{\mu}}, \mathcal{R}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \triangleright_{\mu} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** Completeness is obvious because  $\mathcal{P} - \mathcal{P}_{\pi, \triangleright_{\mu}} \subseteq \mathcal{P}$ . For soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$  but there is no infinite minimal  $(\mathcal{P} - \mathcal{P}_{\pi, \triangleright_{\mu}}, \mathcal{R}, \mu)$ -chain. Since  $\mathcal{P}$  is finite, we can assume that there is  $\mathcal{Q} \subseteq \mathcal{P}$  such that  $A$  has a tail  $B$  that is an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mu)$ -chain where all pairs in  $\mathcal{Q}$  are infinitely often used. Assume that  $B$  is as follows (since  $\mathcal{Q}_{\mathcal{N}} = \emptyset$ , we use a simpler notation):

$$t_0 \xrightarrow{*}_{\mathcal{R}, \mu} s_1 \xrightarrow{\Delta}_{\mathcal{Q}, \mu} t_1 \xrightarrow{*}_{\mathcal{R}, \mu} s_2 \xrightarrow{\Delta}_{\mathcal{Q}, \mu} t_2 \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

where there is a substitution  $\sigma$  such that, for all  $i \geq 1$ ,  $s_i = \sigma(u_i)$  and  $t_i = \sigma(v_i)$  for some  $u_i \rightarrow v_i \in \mathcal{Q}$ . Furthermore, w.l.o.g. we also assume that  $t_0 = \sigma(v_0)$  for some  $u_0 \rightarrow v_0 \in \mathcal{P}$ .

Note that, for all  $i \geq 1$ ,  $\text{root}(s_i) \in \text{Root}(\mathcal{P})$  because  $\text{root}(u_i) \in \text{Root}(\mathcal{P})$ . Since  $\text{root}(v_i) \notin \mathcal{X}$ , we have that  $\text{root}(v_i) \in \text{Root}(\mathcal{P})$ . Then, for all  $i \geq 0$ ,  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ . Therefore, we can apply  $\pi$  to  $s_{i+1}$  and  $t_i$  for all  $i \geq 0$ . Moreover, since  $t_i \xrightarrow{*}_{\mathcal{R}, \mu} s_{i+1}$  for all  $i \geq 0$  and  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , we can actually write  $t_i \xrightarrow{\Delta}_{\mathcal{R}, \mu} s_{i+1}$  because  $\mu$ -rewritings with  $\mathcal{R}$  cannot change  $\text{root}(t_i)$ . Hence,  $\pi(t_i) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_{i+1})$  and also  $\text{root}(t_i) = \text{root}(s_{i+1})$  for all  $i \geq 0$ . Finally, since  $\pi(u_i) \triangleright_{\mu} \pi(v_i)$  for all  $i \geq 0$ , by stability of  $\triangleright_{\mu}$ , we have

$$\pi(s_i) = \pi(\sigma(u_i)) = \sigma(\pi(u_i)) \triangleright_{\mu} \sigma(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$$

for all  $i \geq 1$ . No pair  $u \rightarrow v \in \mathcal{Q}$  satisfies that  $\pi(u) \triangleright_{\mu} \pi(v)$ . Otherwise, we get a contradiction in both of the following two complementary cases:

1. If  $\pi(f) \notin \mu(f)$  for all  $f \in \text{Root}(\mathcal{Q})$ , then, for all  $i \geq 0$ ,  $\pi(t_i) = \pi(s_{i+1})$ , because no  $\mu$ -rewritings are possible on the  $\pi(\text{root}(t_i))$ th immediate subterm  $\pi(t_i)$  of  $t_i$ . Since  $\pi(s_{i+1}) \supseteq_{\mu} \pi(t_{i+1})$ , we have that  $\pi(t_i) \supseteq_{\mu} \pi(t_{i+1})$  for all  $i \geq 0$ . Furthermore, since we assume  $\pi(u) \supseteq_{\mu} \pi(v)$  for some  $u \rightarrow v \in \mathcal{Q}$  which occurs infinitely often in  $B$ , and by stability of  $\supseteq_{\mu}$ , there is a maximal infinite set  $J = \{j_1, j_2, \dots\} \subseteq \mathbb{N}$  such that  $\pi(t_{j_i}) \supseteq_{\mu} \pi(t_{j_{i+1}})$  for all  $i \geq 1$ . We obtain an infinite sequence  $\pi(t_{j_1}) \supseteq_{\mu} \pi(t_{j_2}) \supseteq_{\mu} \dots$  which contradicts the well-foundedness of  $\supseteq_{\mu}$ .
2. If  $\pi(f) \in \mu(f)$  for some  $f \in \text{Root}(\mathcal{Q})$ , then, since  $\text{root}(t_i) = \text{root}(s_{i+1})$  and all pairs in  $\mathcal{Q}$  occur infinitely often in  $B$ , we can assume that  $\text{root}(t_0) = f$ . Furthermore, since  $A$  is minimal, we can assume that  $t_0$  is  $\mu$ -terminating (with respect to  $\mathcal{R}$ ). Since  $\pi(t_i) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_{i+1})$  and  $\pi(s_{i+1}) \supseteq_{\mu} \pi(t_{i+1})$  for all  $i \geq 0$ , the sequence  $B$  is transformed into an infinite  $\xrightarrow{*}_{\mathcal{R}, \mu} \cup \supseteq_{\mu}$ -sequence

$$\pi(t_0) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_1) \supseteq_{\mu} \pi(t_1) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_2) \supseteq_{\mu} \pi(t_2) \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

containing infinitely many  $\supseteq_{\mu}$ -steps, due to  $\pi(u) \supseteq_{\mu} \pi(v)$  for some  $u \rightarrow v \in \mathcal{Q}$  which occurs infinitely often in  $B$ . Since  $\supseteq_{\mu}$  is well-founded, the infinite sequence must also contain infinitely many  $\xrightarrow{*}_{\mathcal{R}, \mu}$ -steps. By making repeated use of the fact that  $\supseteq_{\mu} \circ \xrightarrow{*}_{\mathcal{R}, \mu} \subseteq \xrightarrow{*}_{\mathcal{R}, \mu} \circ \supseteq_{\mu}$ , we obtain an infinite  $\xrightarrow{*}_{\mathcal{R}, \mu}$ -sequence starting from  $\pi(t_0)$ . Thus,  $\pi(t_0)$  is not  $\mu$ -terminating with respect to  $\mathcal{R}$ . Since  $\pi(f) \in \mu(f)$  and hence  $t_0 \supseteq_{\mu} \pi(t_0)$ , this implies that  $t_0$  is not  $\mu$ -terminating (use Lemma 1(1)). This contradicts  $\mu$ -termination of  $t_0$ .

Hence,  $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_{\pi, \supseteq_{\mu}}$  and  $B$  is an infinite minimal  $(\mathcal{P} - \mathcal{P}_{\pi, \supseteq_{\mu}}, \mathcal{R}, \mu)$ -chain. This contradicts our initial argument.  $\square$

**Example 22** (Proof of termination of the main example). Consider the termination problems obtained in Example 15 for the CS-TRS in Example 1:

$$\tau_1 = (\{(1)\}, \mathcal{R}, \mu^{\sharp}), \quad \tau_2 = (\{(17)\}, \mathcal{R}, \mu^{\sharp}), \quad \tau_3 = (\{(21)\}, \mathcal{R}, \mu^{\sharp}), \quad \text{and} \quad \tau_4 = (\{(23)\}, \mathcal{R}, \mu^{\sharp})$$

We apply  $\text{Proc}_{\text{SubNColl}}$  to all such problems. For  $\tau_1$ , with  $\pi(\text{ADD}) = 1$ , we have  $\pi(\text{ADD}(s(n), m)) = s(n) \supseteq_{\mu} n = \pi(\text{ADD}(n, m))$ . Now,  $\text{Proc}_{\text{SubNColl}}(\tau_1) = \{(\emptyset, \mathcal{R}, \mu^{\sharp})\}$ . With  $\text{Proc}_{\text{Fin}}$  we conclude that  $\tau_1$  is finite. Since this can be done for  $\tau_2, \tau_3$ , and  $\tau_4$ , the  $\mu$ -termination of  $\mathcal{R}$  is proved.

The following examples show that if  $\mathcal{P}$  contains collapsing rules, then Theorem 12 does not hold.

**Example 23.** Consider the two TRSs

$$\mathcal{R} : h(x) \rightarrow \varepsilon(g(h(x))) \quad \text{and} \quad \mathcal{P} : \varepsilon(g(x)) \rightarrow x$$

Let  $\mu$  be given by  $\mu(f) = \{1, \dots, k\}$  for all symbols  $f$ . Note that,  $\text{Root}(\mathcal{P}) = \{\varepsilon\}$  and  $\mathcal{D} = \{h\}$  are disjoint. By using the projection  $\pi(\varepsilon) = 1$ , we get  $\pi(\varepsilon(g(x))) = g(x) \supseteq_{\mu} x$ . After removing the pair in  $\mathcal{P}$ , a finite CS problem  $(\emptyset, \mathcal{R}, \mu)$  is obtained. However,  $(\mathcal{P}, \mathcal{R}, \mu)$  is not finite:

$$\varepsilon(g(h(x))) \xrightarrow{*}_{\mathcal{P}, \mu} h(x) \xrightarrow{*}_{\mathcal{R}, \mu} \varepsilon(g(h(x))) \xrightarrow{*}_{\mathcal{P}, \mu} \dots$$

In the following theorem, we show how to use the subterm criterion to remove all collapsing pairs from  $\mathcal{P}$ .

**Theorem 13** (Subterm processor for collapsing pairs). Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs such that  $\mathcal{P}_{\mathcal{G}}$  contains no collapsing rule,  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$  such that

1.  $\pi(\mathcal{P}) \subseteq \supseteq_{\mu}$ , and
2. whenever  $\mathcal{P}_x \neq \emptyset$ , we have  $\pi(f^{\sharp}) \in \mu(f^{\sharp}) \cap \mu(f)$  for all  $f \in \mathcal{H}_{\mathcal{P}}$ .

Then, the processor  $\text{Proc}_{\text{SubColl}}$  given by

$$\text{Proc}_{\text{SubColl}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)\} & \text{if (1) and (2) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** Completeness is obvious because  $\mathcal{P}_{\mathcal{G}} \subseteq \mathcal{P}$ . For soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$  but there is no infinite minimal  $(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)$ -chain. Since  $\mathcal{P}$  is finite, we can assume that

there is  $Q \subseteq \mathcal{P}$  such that  $A$  has a tail  $B$  which is an infinite minimal  $(Q, \mathcal{R}, \mu)$ -chain where all pairs in  $Q$  are infinitely often used and  $Q$  contains some collapsing pair  $u \rightarrow x \in Q_{\mathcal{X}}$ . Assume that  $B$  is

$$t_0 \xrightarrow{*}_{\mathcal{R}, \mu} s_1 \xrightarrow{\Lambda}_{Q, \mu} \circ \supseteq_{\mu}^{\#} t_1 \xrightarrow{*}_{\mathcal{R}, \mu} s_2 \xrightarrow{\Lambda}_{Q, \mu} \circ \supseteq_{\mu}^{\#} t_2 \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

where there is a substitution  $\sigma$  such that, for all  $i \geq 1$ ,  $s_i = \sigma(u_i)$  for some  $u_i \rightarrow v_i \in \mathcal{P}$ , and

1. if  $v_i \notin \mathcal{X}$ , then  $t_i = \sigma(v_i)$ ,
2. if  $v_i = x_i \in \mathcal{X}$ , then  $x_i \notin \text{Var}^{\mu}(u_i)$  and  $t_i = r_i^{\#}$  for some  $r_i \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that  $\sigma(x_i) = C_i[r_i]_{p_i}$  for some  $C_i[\ ]_{p_i}$  and  $p_i \in \text{Pos}^{\mu}(C_i[\ ]_{p_i})$  such that  $\text{sprex}_{C_i[\ ]_{p_i}}(p_i) \subseteq \mathcal{F}$  and  $r_i = \theta_i(\bar{r}_i)$  for some  $\bar{r}_i \in \mathcal{N}^{\text{HT}}_{\mathcal{Q}}$  and substitution  $\theta_i$ .

Since we can freely choose the starting term of  $B$ , w.l.o.g. we assume that  $t_0$  is a particular case of the second alternative above, i.e., there is a collapsing pair  $u_0 \rightarrow x_0$  such that  $\sigma(x_0) \supseteq_{\mu} r_0$  and  $t_0 = r_0^{\#}$ . Note that, for all  $i \geq 1$ ,  $\text{root}(s_i) \in \text{Root}(\mathcal{P})$  because  $\text{root}(u_i) \in \text{Root}(\mathcal{P})$ . Furthermore, for all  $i \geq 0$ ,  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$  because:

1. If  $u_i \rightarrow v_i \in Q_{\mathcal{G}}$ , then  $\text{root}(v_i) \in \text{Root}(\mathcal{P})$  and  $t_i = \sigma(v_i)$ .
2. If  $u_i \rightarrow v_i \in Q_{\mathcal{X}}$ , then  $\text{root}(t_i) \in \mathcal{D}^{\#}$ ; since  $t_i \xrightarrow{*}_{\mathcal{R}, \mu} s_{i+1}$  and  $\mathcal{D}^{\#} \cap \mathcal{F} = \emptyset$  (see Remark 4), rewritings with  $\mathcal{R}$  cannot remove the marked root symbol in  $t_i$ ; hence, we can further conclude  $\text{root}(t_i) = \text{root}(s_{i+1}) \in \text{Root}(\mathcal{P})$ .

Therefore, we can apply  $\pi$  to  $s_{i+1}$  and  $t_i$  for all  $i \geq 0$ . Moreover, since  $t_i \xrightarrow{*}_{\mathcal{R}, \mu} s_{i+1}$  for all  $i \geq 0$  and  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , we can actually write  $t_i \xrightarrow{\supseteq_{\mu}^{\#}}_{\mathcal{R}, \mu} s_{i+1}$ . Hence,  $\pi(t_i) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_{i+1})$  and also  $\text{root}(t_i) = \text{root}(s_{i+1})$  for all  $i \geq 0$ .

Since  $u \rightarrow x \in Q_{\mathcal{X}}$  and  $B$  is infinite, it must be  $\mathcal{H}_{\mathcal{Q}} \neq \emptyset$  (hence  $\mathcal{H}_{\mathcal{P}} \neq \emptyset$ ). Thus, we have  $\pi(f^{\#}) \in \mu(f)$  for all  $f \in \mathcal{H}_{\mathcal{Q}} \subseteq \mathcal{H}_{\mathcal{P}}$ . Then, since  $\text{root}(t_i) = \text{root}(s_{i+1})$  and all pairs in  $Q$  occur infinitely often in  $B$ , we can assume that  $\text{root}(t_0) = f$ . Furthermore, since  $A$  is minimal, we can assume that  $t_0$  is  $\mu$ -terminating. We have that  $\pi(u_i) \supseteq_{\mu} \pi(v_i)$  for all  $u_i \rightarrow v_i \in Q$ . Now we distinguish two cases:

1. If  $u_i \rightarrow v_i \in Q_{\mathcal{G}}$ , then  $s_i = \sigma(u_i)$  and  $t_i = \sigma(v_i)$ . By stability of  $\supseteq_{\mu}$  we have  $\pi(s_i) \supseteq_{\mu} \pi(t_i)$ .
2. If  $u_i \rightarrow v_i = u_i \rightarrow x_i \in Q_{\mathcal{X}}$ , then  $s_i = \sigma(u_i)$  and there is a term  $r_i$ , such that  $\sigma(x_i) \supseteq_{\mu} r_i$  and  $r_i^{\#} = t_i$ . Since  $\pi(u_i) \supseteq_{\mu} x_i$ , by stability of  $\supseteq_{\mu}$  we have

$$\pi(s_i) = \pi(\sigma(u_i)) = \sigma(\pi(u_i)) \supseteq_{\mu} \sigma(x_i) \supseteq_{\mu} r_i.$$

Note that  $f_i = \text{root}(r_i) = \text{root}(\bar{r}_i) \in \mathcal{H}_{\mathcal{P}}$ . Since  $\pi(t_{i+1}) = t_i|_{\pi(f_i^{\#})} = r_i^{\#}|_{\pi(f_i^{\#})} = r_i|_{\pi(f_i^{\#})}$  and  $\pi(f_i^{\#}) \in \mu(f_i)$ , we have that  $r_i \supseteq_{\mu} \pi(t_i)$  and thus  $\pi(s_i) \supseteq_{\mu} \pi(t_i)$ .

Therefore, by applying the simple projection  $\pi$ , the sequence  $B$  is transformed into an infinite  $\xrightarrow{*}_{\mathcal{R}, \mu} \cup \supseteq_{\mu}$ -sequence  $B'$

$$\pi(t_0) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_1) \supseteq_{\mu} \pi(t_1) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(s_2) \supseteq_{\mu} \pi(t_2) \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

Since  $u \rightarrow x$  occurs infinitely often in  $B$ , and by case (2) above,  $B'$  contains infinitely many  $\supseteq_{\mu}$  steps, starting from  $\pi(t_0)$ . Since  $\supseteq_{\mu}$  is well-founded, the infinite sequence must also contain infinitely many  $\xrightarrow{*}_{\mathcal{R}, \mu}$ -steps. By making repeated use of the fact that  $\supseteq_{\mu} \circ \xrightarrow{*}_{\mathcal{R}, \mu} \subseteq \xrightarrow{*}_{\mathcal{R}, \mu} \circ \supseteq_{\mu}$ , we obtain an infinite  $\xrightarrow{*}_{\mathcal{R}, \mu}$ -sequence starting from  $\pi(t_0)$ . Thus,  $\pi(t_0)$  is not  $\mu$ -terminating with respect to  $\mathcal{R}$ . Since  $\pi(f^{\#}) \in \mu(f^{\#})$  and hence  $t_0 \supseteq_{\mu} \pi(t_0)$ , this implies that  $t_0$  is not  $\mu$ -terminating (use Lemma 1(1)). This contradicts  $\mu$ -termination of  $t_0$ . Therefore,  $Q$  cannot contain collapsing pairs. This contradicts our initial assumption  $u \rightarrow x \in Q$ .  $\square$

**Remark 13.** The use of Theorem 13 only makes sense if  $\mathcal{P} \subseteq \mathcal{P}_{\mathcal{G}} \cup \mathcal{P}_{\mathcal{X}}^1$ . If  $u \rightarrow x \in \mathcal{P}_{\mathcal{X}} - \mathcal{P}_{\mathcal{X}}^1$  for some  $u = f(u_1, \dots, u_k)$ , then for all  $i \in \{1, \dots, k\}$ , whenever  $x \in \text{Var}(u_i)$  we have  $i \in \mu(f)$  and  $u_i \supseteq_{\mu} x$ . Thus, there is no simple projection  $\pi$  such that  $\pi(u) \supseteq_{\mu} x$ .

**Example 24.** Consider the following TRS  $\mathcal{R}$ :

$$\begin{aligned} g(x, y) &\rightarrow f(x, y) \\ f(c(x), y) &\rightarrow g(x, g(y, y)) \end{aligned}$$

together with the replacement map  $\mu$  given by  $\mu(c) = \mu(g) = \{1\}$  and  $\mu(f) = \emptyset$ . The CSDPs are:

$$\mathbb{G}(x, y) \rightarrow \mathbb{F}(x, y) \quad (33)$$

$$\mathbb{F}(c(x), y) \rightarrow \mathbb{G}(x, g(y, y)) \quad (34)$$

$$\mathbb{F}(c(x), y) \rightarrow x \quad (35)$$

and all of them are part of the only SCC  $\mathcal{P} = \{(33), (34), (35)\}$  in the CSDG of  $(\mathcal{R}, \mu)$ . Note that  $\mathcal{NHT}_{\mathcal{P}} = \{g(y, y)\}$ ; hence  $\mathcal{H}_{\mathcal{P}} = \{g\}$ . Consider the simple projection  $\pi$  given by  $\pi(\mathbb{F}) = \pi(\mathbb{G}) = 1$ . Note that  $\pi(\mathbb{G}) \in \mu^{\sharp}(\mathbb{G}) \cap \mu^{\sharp}(g)$  as required by Theorem 13. We have

- $\pi(\mathbb{G}(x, y)) = x \triangleright_{\mu} x = \pi(\mathbb{F}(x, y))$
- $\pi(\mathbb{F}(c(x), y)) = c(x) \triangleright_{\mu} x = \pi(\mathbb{G}(x, g(y, y)))$ ,
- $\pi(\mathbb{F}(c(x), y)) = c(x) \triangleright_{\mu} x = \pi(x)$

We use  $\text{Proc}_{\text{subColl}}$  to remove (35) from  $\mathcal{P}$  and obtain a new problem  $(\{(33), (34)\}, \mathcal{R}, \mu^{\sharp})$ . Then,  $\text{Proc}_{\text{subColl}}$  applies and yields  $(\{(33)\}, \mathcal{R}, \mu^{\sharp})$ . With  $\text{Proc}_{\text{SCC}}$ , we conclude the  $\mu$ -termination of  $\mathcal{R}$ .

The following result provides a kind of generalization of the subterm criterion to simple projections that only take non- $\mu$ -replacing arguments.

**Theorem 14** (Non- $\mu$ -replacing projection processor). *Let  $\mathcal{R} = (\mathcal{F}, R) = (C \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (G, P)$  be TRSs such that  $\mathcal{P}_{\mathbb{G}}$  contains no collapsing rule,  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded and  $\pi$  be a simple projection for  $\mathcal{P}$  such that*

1. for all  $f \in \text{Root}(\mathcal{P})$ ,  $\pi(f) \notin \mu(f)$ ,
2.  $\pi(\mathcal{P}) \subseteq \succsim$ ,
3. whenever  $\mathcal{NHT}_{\mathcal{P}} \neq \emptyset$  and  $\mathcal{P}_x \neq \emptyset$ , we have that  $\text{Emb}^{\mu}(\mathcal{F}) \subseteq \succsim$  and  $t \succsim t|_{\pi(\text{root}(t)^{\sharp})}$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ .

Let  $\mathcal{P}_{>} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$ . Then, the processor  $\text{Proc}_{\text{NRP}}$  given by

$$\text{Proc}_{\text{NRP}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P} - \mathcal{P}_{>}, \mathcal{R}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

**Proof.** Completeness is obvious because  $\mathcal{P} - \mathcal{P}_{>} \subseteq \mathcal{P}$ . For soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain  $A$  but there is no infinite minimal  $(\mathcal{P} - \mathcal{P}_{>}, \mathcal{R}, \mu)$ -chain. Since  $\mathcal{P}$  is finite, we can assume that there is  $Q \subseteq \mathcal{P}$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \xrightarrow{\Delta}_{Q, \mu} \circ \triangleright_{\mu}^{\sharp} t_1 \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2) \xrightarrow{\Delta}_{Q, \mu} \circ \triangleright_{\mu}^{\sharp} t_2 \xrightarrow{*}_{\mathcal{R}, \mu} \dots$$

for some substitution  $\sigma$  and pairs  $u_i \rightarrow v_i \in Q$ , and

1. if  $v_i \notin \mathcal{X}$ , then  $t_i = \sigma(v_i)$ , and
2. if  $v_i = x_i \in \mathcal{X}$ , then  $x_i \notin \text{Var}^{\mu}(u_i)$  and  $t_i = s_i^{\sharp}$  for some  $s_i$  such that  $\sigma(x_i) = C_i[s_i]_{p_i}$  for some  $C_i[ ]_{p_i}$  and  $p_i \in \text{Pos}^{\mu}(C_i[ ]_{p_i})$  such that  $\text{sprefix}_{C_i[ ]_{p_i}}(p_i) \subseteq \mathcal{F}$  and  $s_i = \theta_i(\bar{s}_i)$  for some  $\bar{s}_i \in \mathcal{NHT}_{\mathcal{P}}$  and substitution  $\theta_i$ .

Furthermore, all pairs in  $Q$  are used infinitely often in  $B$ . As discussed in the proof of Theorem 12, for all  $i \geq 1$ ,  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ ,  $\pi(t_i) \xrightarrow{*}_{\mathcal{R}, \mu} \pi(\sigma(u_{i+1}))$  and also  $\text{root}(t_i) = \text{root}(u_{i+1})$  for all  $i \geq 1$ . No pair  $u \rightarrow v \in Q$  satisfies that  $\pi(u) > \pi(v)$ . Otherwise, by applying the simple projection  $\pi$  to the sequence  $B$ , we get a contradiction as follows:

1. Since  $\pi(f) \notin \mu(f)$  for all  $f \in \text{Root}(Q)$ , no  $\mu$ -rewritings are possible on the subterm  $\pi(t_i)$  of  $t_i$ . Therefore, for all  $i \geq 1$ ,  $\pi(t_i) = \pi(\sigma(u_{i+1})) = \sigma(\pi(u_{i+1}))$ .
2. Due to  $\pi(u_i) \succsim \pi(v_i)$  and by stability of  $\succsim$ , we have that  $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \succsim \sigma(\pi(v_i))$ . Now, we distinguish two cases:
  - (a) If  $u_i \rightarrow v_i \in Q_{\mathcal{G}}$ , then  $\pi(t_i) = \pi(\sigma(v_i)) = \sigma(\pi(v_i))$ . Thus,  $\pi(\sigma(u_i)) \succsim \pi(t_i)$ .
  - (b) If  $u_i \rightarrow v_i \in Q_{\mathcal{X}}$ , then  $\sigma(\pi(v_i)) = \sigma(x_i)$ . We have that  $\sigma(x_i) \succsim s_i$  (because  $\text{Emb}^{\mu}(\mathcal{F}) \subseteq \succsim$ ). Let  $f = \text{root}(u_{i+1}) = \text{root}(t_i) = \text{root}(s_i^{\sharp})$ . Since  $t \succsim t|_{\pi(\text{root}(t)^{\sharp})}$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ , by stability, we have  $s_i = \theta_i(\bar{s}_i) \succsim \theta_i(\bar{s}_i|_{\pi(f)}) = \theta_i(\bar{s}_i)|_{\pi(f)} = s_i|_{\pi(f)}$ . Since  $s_i|_{\pi(f)} = t_i|_{\pi(f)} = \pi(t_i)$ , we have  $s_i \succsim \pi(t_i)$ . Hence,  $\pi(\sigma(u_i)) \succsim \pi(t_i)$ .

Thus, we always have  $\pi(\sigma(u_i)) \succsim \pi(t_i)$ . We obtain an infinite  $\succsim$  sequence

$$\pi(\sigma(u_1)) \succsim \pi(t_1) = \pi(\sigma(u_2)) \succsim \pi(t_2) \cdots$$

Since pairs in  $\mathcal{Q}$  occur infinitely often, this sequence contains infinitely many  $>$  steps starting from  $\pi(\sigma(u_1))$ . This contradicts the well-foundedness of  $>$ .

Therefore,  $\mathcal{Q} \subseteq \mathcal{P} - \mathcal{P}_{>}$ , i.e.,  $B$  is an infinite minimal  $(\mathcal{P} - \mathcal{P}_{>}, \mathcal{R}, \mu)$ -chain. This contradicts our initial assumption.  $\square$

**Example 25.** Consider the CS-TRS  $(\mathcal{R}, \mu)$  in Example 10.  $\text{DP}(\mathcal{R}, \mu)$  is:

$$\mathcal{G}(x) \rightarrow \mathcal{H}(x) \qquad \mathcal{H}(\mathfrak{d}) \rightarrow \mathcal{G}(c)$$

where  $\mu^\sharp(\mathcal{G}) = \mu^\sharp(\mathcal{H}) = \emptyset$ . The dependency graph contains a single cycle that includes both pairs. The only simple projection is  $\pi(\mathcal{G}) = \pi(\mathcal{H}) = 1$ . Since  $\pi(\mathcal{G}(x)) = \pi(\mathcal{H}(x))$ , we only need to guarantee that  $\pi(\mathcal{H}(\mathfrak{d})) = \mathfrak{d} > c = \pi(\mathcal{G}(c))$  holds for a stable and well-founded ordering  $>$  (e.g., an RPO with  $\mathfrak{d} > c$ ).

**Theorem 15** (Non- $\mu$ -replacing projection processor II). *Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs such that  $\mathcal{P}_{\mathcal{G}}$  contains no collapsing rule,  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded and let  $\pi$  be a simple projection for  $\mathcal{P}$  such that*

1. for all  $f \in \text{Root}(\mathcal{P})$ ,  $\pi(f) \notin \mu(f)$ ,
2.  $\pi(\mathcal{P}) \subseteq \succsim$ ,
3. whenever  $\mathcal{NHT}_{\mathcal{P}} \neq \emptyset$  and  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , we have that  $\text{Emb}^{\mu}(\mathcal{F}) \subseteq \succsim$  and  $t > t|_{\pi(\text{root}(t)^\sharp)}$  for all  $t \in \mathcal{NHT}_{\mathcal{P}}$ .

Then, the processor  $\text{Proc}_{\text{NRP2}}$  given by

$$\text{Proc}_{\text{NRP2}}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} \{(\mathcal{P}_{\mathcal{G}}, \mathcal{R}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

## 12. Narrowing transformation

The starting point of a proof of  $\mu$ -termination of a TRS  $\mathcal{R}$  is the computation of the estimated CSDG,  $\text{EDG}(\mathcal{R}, \mu)$ , followed by the use of the SCC processor (Theorem 6). The estimation of the graph can lead to *overestimating* the arcs that connect two CSDPs.

**Example 26.** Consider the following example [50, Proposition 7]:

$$\begin{aligned} \mathfrak{f}(0) &\rightarrow \text{cons}(0, \mathfrak{f}(\mathfrak{s}(0))) & \mathfrak{p}(\mathfrak{s}(x)) &\rightarrow x \\ \mathfrak{f}(\mathfrak{s}(0)) &\rightarrow \mathfrak{f}(\mathfrak{p}(\mathfrak{s}(0))) \end{aligned}$$

together with  $\mu(\mathfrak{f}) = \mu(\mathfrak{p}) = \mu(\mathfrak{s}) = \mu(\text{cons}) = \{1\}$  and  $\mu(0) = \emptyset$ .  $\text{DP}(\mathcal{R}, \mu)$  consists of the pairs:

$$\mathfrak{F}(\mathfrak{s}(0)) \rightarrow \mathfrak{F}(\mathfrak{p}(\mathfrak{s}(0))) \tag{36}$$

$$\mathfrak{F}(\mathfrak{s}(0)) \rightarrow \mathfrak{P}(\mathfrak{s}(0)) \tag{37}$$

The *estimated* CS-dependency graph contains one cycle:  $\{(36)\}$ . However, this cycle does *not* belong to the CS-dependency graph because there is no way to  $\mu$ -rewrite  $\mathfrak{F}(\mathfrak{p}(\mathfrak{s}(0)))$  into  $\mathfrak{F}(\mathfrak{s}(0))$ .

As already observed by Arts and Giesl for the standard case [10], in our case, the overestimation comes when a (non-collapsing) pair  $u_i \rightarrow v_i$  is followed in a chain by a second one  $u_{i+1} \rightarrow v_{i+1}$  and  $v_i$  and  $u_{i+1}$  are not directly unifiable, i.e., at least one  $\mu$ -rewriting step is needed to  $\mu$ -reduce  $\sigma(v_i)$  to  $\sigma(u_{i+1})$ . Then, we always have  $\sigma(v_i) \xrightarrow{\mathcal{R}, \mu^\sharp} \sigma(v'_i) \xrightarrow{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})$ . Then,  $v'_i$  is a one-step  $\mu$ -narrowing of  $v_i$ , and we could require  $u_i \sqsupset v'_i$  (which could be easier to prove) instead of  $u_i \sqsupset v_i$ . Furthermore, we could discover that  $v_i$  has no  $\mu$ -narrowings. In this case, we know that no chain starts from  $\sigma(v_i)$ .

We can be more precise when connecting two pairs  $u \rightarrow v$  and  $u' \rightarrow v'$  in a chain if we perform all the possible one-step  $\mu$ -narrowings on  $v$  in order to develop the possible reductions from  $\sigma(v)$  to  $\sigma(u')$ . Then, we obtain new terms  $v_1, \dots, v_n$ , which are one-step  $\mu$ -narrowings of  $v$  using unifiers  $\theta_i$  (i.e.,  $v \xrightarrow{\mathcal{R}, \mu, \theta_i} v_i$ ) for  $i \in \{1, \dots, n\}$ , respectively. These unifiers are also applied to the left-hand side  $u$  of the pair  $u \rightarrow v$ . Therefore, we can replace a pair  $u \rightarrow v$  by all its (one-step)  $\mu$ -narrowed pairs  $\theta_1(u) \rightarrow v_1, \dots, \theta_n(u) \rightarrow v_n$ .

As in [10,35], a pair  $u \rightarrow v \in \mathcal{P}$  can only be replaced by its  $\mu$ -narrowings if the right-hand side  $v$  does not unify with any left-hand side  $u'$  of a (possibly renamed) pair  $u' \rightarrow v' \in \mathcal{P}$  (note that this excludes pairs  $u \rightarrow v$  with  $v \in \mathcal{X}$ ). Moreover, the term  $v$  must be *linear*. We need to demand linearity instead of (the apparently more natural)  $\mu$ -linearity (i.e., something like “no multiple  $\mu$ -replacing occurrences of the same variable are allowed”).

**Example 27.** The following TRS is used in [10] to motivate the requirement of linearity.

$$\begin{aligned} \mathbb{F}(s(x)) &\rightarrow \mathbb{F}(g(x, x)) \\ g(0, 1) &\rightarrow s(0) \\ 0 &\rightarrow 1 \end{aligned}$$

We make it a CS-TRS by adding a replacement map  $\mu$  given by  $\mu(\mathbb{F}) = \mu(s) = \{1\}$ , and  $\mu(g) = \{2\}$ . The only cycle in the CSDG consists of the CSDP

$$\mathbb{F}(s(x)) \rightarrow \mathbb{F}(g(x, x)).$$

If linearity of the right-hand sides is not required for  $\mu$ -narrowing CSDPs, then this pair will be removed, since  $\mathbb{F}(g(x, x))$  and the (renamed version of) the left-hand side  $\mathbb{F}(s(x'))$  do not unify. Thus, there are no  $\mu$ -narrowings. However, the system is *not*  $\mu$ -terminating:

$$\underline{\mathbb{F}(s(0))} \hookrightarrow \underline{\mathbb{F}(g(0, 0))} \hookrightarrow \underline{\mathbb{F}(g(0, 1))} \hookrightarrow \underline{\mathbb{F}(s(0))} \dots$$

The problem is that the  $\mu$ -reduction from  $\sigma(\mathbb{F}(g(x, x)))$  to  $\sigma(\mathbb{F}(s(x')))$  takes place ‘in  $\sigma$ ’, and, therefore, it cannot be captured by  $\mu$ -narrowing. Note that  $\mathbb{F}(g(x, x))$  is “ $\mu$ -linear”.

Another restriction to take into account when  $\mu$ -narrowing a noncollapsing pair  $u \rightarrow v$  is that the  $\mu$ -replacing variables in  $v$  have to be  $\mu$ -replacing in  $u$  as well (this corresponds with the notion of conservativeness). Furthermore, they cannot be both  $\mu$ -replacing and non- $\mu$ -replacing at the same time. This corresponds to the following definition.

**Definition 14** (Strongly conservative [29]). Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . A rule  $l \rightarrow r$  is strongly  $\mu$ -conservative if it is  $\mu$ -conservative and  $\text{Var}^{\mu}(l) \cap \text{Var}^{\neq \mu}(l) = \text{Var}^{\mu}(r) \cap \text{Var}^{\neq \mu}(r) = \emptyset$ .

The following result shows that, under these conditions, the set of CSDPs can be safely replaced by their  $\mu$ -narrowings.

**Theorem 16** (Narrowing processor). Let  $\mathcal{R}$  and  $\mathcal{P}$  be TRSs and  $\mu \in M_{\mathcal{R} \cup \mathcal{P}}$ . Let  $u \rightarrow v \in \mathcal{P}$  be such that

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

Let  $\mathcal{Q} = (\mathcal{P} - \{u \rightarrow v\}) \cup \{u' \rightarrow v' \mid u' \rightarrow v' \text{ is a } \mu\text{-narrowing of } u \rightarrow v\}$ . Then, the processor  $\text{Proc}_{\text{narr}}$  given by

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

is

1. sound whenever  $u \rightarrow v$  is strongly conservative,
2. complete in all cases.

**Proof.** The proof of this theorem is analogous to the proof of [35, Theorem 31], which we adapt here. For soundness, we prove that given a minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain “ $\dots, u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ ”, there is a  $\mu$ -narrowing  $v'$  of  $v$  with the mgu  $\theta$  such that “ $\dots, u_1 \rightarrow v_1, \theta(u) \rightarrow v', u_2 \rightarrow v_2, \dots$ ” is also a minimal  $(\mathcal{Q}, \mathcal{R}, \mu)$ -chain. Hence, every infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain yields an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mu)$ -chain.

If “ $\dots, u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ ” is a minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain, then there is a substitution  $\sigma$  such that for all pairs  $s \rightarrow t$  in the chain,

1. if  $s \rightarrow t \in \mathcal{P}_{\mathcal{G}}$ , then  $\sigma(t)$  is  $\mu$ -terminating and it  $\mu$ -reduces to the instantiated left-hand side  $\sigma(s')$  of the next pair  $s' \rightarrow t'$  in the chain
2. if  $s \rightarrow t = s \rightarrow x \in \mathcal{P}_{\mathcal{X}}$  then,  $\sigma(x)$  has a  $\mu$ -replacing subterm  $s_0$ ,  $\sigma(x) \succeq_{\mu} s_0$  such that  $s_0^{\#}$  is  $\mu$ -terminating and it  $\mu$ -reduces to the instantiated left-hand side  $\sigma(s')$  of the next pair  $s' \rightarrow t'$  in the chain; furthermore, there is  $\bar{s}_0 \in \mathcal{NHT}(\mathcal{R}, \mu)$  such that  $s_0 = \theta_0(\bar{s}_0)$  for some substitution  $\theta_0$ .

Assume that  $\sigma$  is a substitution satisfying the above requirements and such that the length of the sequence  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2)$  is *minimum*. Note that the length of this  $\mu$ -reduction sequence cannot be zero because  $v$  and  $u_2$  do not unify, that is,  $\sigma(v) \neq \sigma(u_2)$ . Hence, there is a term  $q$  such that  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} q \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2)$ . We consider two possible cases:

1. The reduction  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} q$  takes place within a binding of  $\sigma$ , i.e., there is a term  $r$ , a  $\mu$ -replacing variable position  $p \in \mathcal{P}os_{\mathcal{X}}^{\mu}(v)$ , and a  $\mu$ -replacing variable  $x \in \mathcal{V}ar^{\mu}(v)$  such that  $v|_p = x$ ,  $q = \sigma(v[r]_p)$  and  $\sigma(x) \xrightarrow{*}_{\mathcal{R}, \mu} r$ . Since  $v$  is linear,  $x$  occurs only once in  $v$ . Thus,  $q = \sigma'(v)$  for the substitution  $\sigma'$  with  $\sigma'(x) = r$  and  $\sigma'(y) = \sigma(y)$  for all variables  $y \neq x$ . As we assume that all occurrences of pairs in the chain are variable disjoint,  $\sigma'(x)$  behaves like  $\sigma$  for all pairs except  $u \rightarrow v$ . We have  $\sigma(z) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma'(z)$  for all  $z \in \mathcal{X}$ . Since  $u \rightarrow v$  is strongly conservative, we also have  $\sigma(u) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma'(u)$  because all occurrences of  $x$  in  $u$  must be  $\mu$ -replacing. Hence, if  $u_1 \rightarrow v_1 \in \mathcal{P}_G$  we have

$$\sigma'(v_1) = \sigma(v_1) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma'(u)$$

and if  $u_1 \rightarrow v_1 \in \mathcal{P}_{\mathcal{X}}$ , then there is  $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that

$$\sigma'(v_1) = \sigma(v_1) \triangleright_{\mu} s_1 \text{ and } s_1^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma'(u)$$

and, in both cases,

$$\sigma'(v) = q \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2) = \sigma'(u_2).$$

Note that, by minimality and because  $u \rightarrow v \in \mathcal{P}_G$ ,  $\sigma(v)$  is  $(\mathcal{R}, \mu)$ -terminating and, since  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} q$ , the term  $q$  is  $(\mathcal{R}, \mu)$ -terminating as well. Therefore,  $\sigma'(x) = q$  is  $(\mathcal{R}, \mu)$ -terminating and  $\sigma'$  satisfies the two conditions above. Since the length of the sequence  $\sigma'(v) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma'(u_2)$  is shorter than the sequence  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2)$ , we obtain a contradiction and we conclude that the  $\mu$ -reduction  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} q$  cannot take place in a binding of  $\sigma$ .

2. The reduction  $\sigma(v) \xrightarrow{*}_{\mathcal{R}, \mu} q$  'touches'  $v$ , i.e., there is a nonvariable position  $p \in \mathcal{P}os_{\mathcal{F}}^{\mu}(v)$ , and a rewrite rule  $l \rightarrow r \in \mathcal{R}$  such that  $\sigma(v|_p) = \rho(l)$ , for some substitution  $\rho$  and

$$\sigma(v) = \sigma(v)[\sigma(v|_p)]_p = \sigma(v)[\rho(l)]_p \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(v)[\rho(r)]_p = q$$

Since we can assume that variables in  $l$  are fresh, we can extend  $\sigma$  to behave like  $\rho$  on variables in  $l$ . Thus,  $\sigma(l) = \sigma(v|_p)$ , i.e.,  $l$  and  $v|_p$  unify and there is a mgu  $\theta$  and a substitution  $\tau$  satisfying  $\sigma(x) = \tau(\theta(x))$  for all variables  $x$ . We have that  $v \mu$ -narrows to  $\theta(v)[\theta(r)]_p = v'$  with unifier  $\theta$ . Again, we can extend  $\sigma$  to behave like  $\tau$  on the variables of  $\theta(u)$  and  $v'$ . Therefore, if  $u_1 \rightarrow v_1 \in \mathcal{P}_G$  we have

$$\sigma(v_1) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u) = \tau(\theta(u)) = \sigma(\theta(u))$$

and if  $u_1 \rightarrow v_1 \in \mathcal{P}_{\mathcal{X}}$ , then there is  $s_1 \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  such that

$$\sigma(v_1) = \sigma(x) \triangleright_{\mu} s_1 \text{ and } s_1^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u) = \tau(\theta(u)) = \sigma(\theta(u))$$

and

$$\sigma(v') = \tau(v') = \tau(\theta(v))[\tau(\theta(r))]_p = \sigma(v)[\sigma(r)]_p = \sigma(v)[\rho(r)]_p = q \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2)$$

Hence, "...  $u_1 \rightarrow v_1, \theta(u) \rightarrow v', u_2 \rightarrow v_2, \dots$ " is also a minimal chain.

Completeness is also analogous to the 'completeness' part of [35, Theorem 31]. If  $(Q, \mathcal{R}, \mu)$  is infinite and  $\mathcal{R}$  is non- $\mu$ -terminating, then  $(\mathcal{P}, \mathcal{R}, \mu)$  is infinite as well. If  $\mathcal{R}$  is  $\mu$ -terminating, then let "...  $u_1 \rightarrow v_1, \theta(u) \rightarrow v', u_2 \rightarrow v_2, \dots$ " be an infinite minimal  $(Q, \mathcal{R}, \mu)$ -chain where  $v'$  is a one-step  $\mu$ -narrowing of  $v$  using the mgu  $\theta$ . We prove that "...  $u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ " is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain. There is a substitution  $\sigma$  such that

$$\sigma(v_1) \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(\theta(u)) \text{ if } u_1 \rightarrow v_1 \in \mathcal{P}_G, \text{ and}$$

$$\sigma(v_1) = \sigma(x) \triangleright_{\mu} s_1 \text{ and } s_1^{\sharp} \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(\theta(u)) \text{ if } u_1 \rightarrow v_1 \in \mathcal{P}_{\mathcal{X}}$$

Finally, we also have

$$\sigma(v') \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_2).$$

Since the variables in the pairs are pairwise disjoint, we may extend  $\sigma$  to behave like  $\sigma(\theta(x))$  on  $x \in \text{Var}(u)$  then  $\sigma(u) = \sigma(\theta(u))$  and therefore

$$\begin{aligned} \sigma(v_1) &\hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u) && \text{if } u_1 \rightarrow v_1 \in \mathcal{P}_G, \text{ and} \\ \sigma(v_1) &\succeq_{\mu} s_1 \text{ and } s_1^{\sharp} && \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u) \text{ if } u_1 \rightarrow v_1 \in \mathcal{P}_X \end{aligned}$$

Moreover, by definition of  $\mu$ -narrowing, we have  $\theta(v) \hookrightarrow_{\mathcal{R}, \mu} v'$ . This implies that  $\sigma(\theta(v)) \hookrightarrow_{\mathcal{R}, \mu} \sigma(v')$ , and since  $\sigma(v) = \sigma(\theta(v))$ , we obtain

$$\sigma(v) \hookrightarrow_{\mathcal{R}, \mu} \sigma(v') \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_2).$$

Since  $\mathcal{R}$  is  $\mu$ -terminating,  $\sigma(v)$  is  $(\mathcal{R}, \mu)$ -terminating. Hence, “. . . ,  $u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \dots$ ” is a minimal infinite  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain as well.  $\square$

**Example 28.** Since the right-hand side of pair (36) in Example 26 does not unify with any (renamed) left-hand side of a CSDP (including itself) and it can be  $\mu$ -narrowed at position 1 (notice that  $\mu(\varepsilon) = \{1\}$ ) by using the rule  $\mathcal{P}(\varepsilon(x)) \rightarrow x$ , we can replace it by its  $\mu$ -narrowed pair:

$$\mathcal{F}(\varepsilon(0)) \rightarrow \mathcal{F}(0) \tag{38}$$

Now,  $\text{Proc}_{\text{SCC}}(\{(38)\}, \mathcal{R}, \mu) = \emptyset$  and the  $\mu$ -termination of  $\mathcal{R}$  is proved.

The following example shows that strong conservativeness cannot be dropped for the pair  $u \rightarrow v$  to be  $\mu$ -narrowed. This requirement was not taken into account in [4, Theorem 5.3].

**Example 29.** Consider the following<sup>8</sup> TRS  $\mathcal{R}$ :

$$\begin{aligned} c(e(x)) &\rightarrow d(x, x) \\ a &\rightarrow e(a) \end{aligned}$$

and  $\mathcal{P}$  consisting of the following pair:

$$\mathcal{F}(d(x, x)) \rightarrow \mathcal{F}(c(x))$$

together with  $\mu(c) = \mu(d) = \mu(\mathcal{F}) = \{1\}$  and  $\mu(e) = \emptyset$ . There is an infinite  $(\mathcal{P}, \mathcal{R}, \mu)$ -chain:

$$\mathcal{F}(c(\underline{a})) \hookrightarrow_{\mathcal{R}, \mu} \mathcal{F}(c(e(\underline{a}))) \hookrightarrow_{\mathcal{R}, \mu} \mathcal{F}(d(\underline{a}, \underline{a})) \hookrightarrow_{\mathcal{P}, \mu} \mathcal{F}(c(\underline{a})) \hookrightarrow_{\mathcal{R}, \mu} \dots$$

Since  $\mathcal{F}(c(x))$  does not unify with any left-hand side of another pair, we can  $\mu$ -narrow the pair in  $\mathcal{P}$ . We obtain  $\mathcal{P}'$  consisting of the  $\mu$ -narrowed pair

$$\mathcal{F}(d(e(x), e(x))) \rightarrow \mathcal{F}(d(x, x))$$

No infinite  $(\mathcal{P}', \mathcal{R}, \mu)$ -chain is possible now. Note that  $\mathcal{P}$  is  $\mu$ -conservative, but it is *not* strongly  $\mu$ -conservative (the variable  $x$  is both  $\mu$ -replacing and non- $\mu$ -replacing in  $\mathcal{F}(d(x, x))$ ).

**Remark 14** (Implementing the narrowing processor). In our current implementation, we apply the narrowing processor only if, after computing the (one-step)  $\mu$ -narrowings of the right-hand side  $v$  of a pair  $u \rightarrow v \in \mathcal{P}$ , the new CS-dependency graph does not increase the number of arcs. More sophisticated strategies like (the corresponding adaptations of) the *safe transformations* in [35, Definition 33] could be considered in the future.

### 13. Experiments

The processors described in the previous sections were implemented as part of the tool `MU-TERM`. We tested the CSDP-framework in practice on the 90 examples in the Context-Sensitive Rewriting subcategory of the 2007 International Termination Competition:

[http : //www.lri.fr/~marche/termination – competition/2007/](http://www.lri.fr/~marche/termination-competition/2007/).

These 90 examples are part of the Termination Problem Data Base (TPDB, version 4.0):

[http : //www.lri.fr/~marche/tpdb/](http://www.lri.fr/~marche/tpdb/).

<sup>8</sup> We thank Fabian Emmes for providing this example.

**Table 1**  
Comparison among CSR termination techniques.

Tool version	Proved	Total time	Average time
CSDPs	65/90	0.31 s	0.00 s
CSRPO	37/90	0.21 s	0.00 s
Polynomial orderings	27/90	0.06 s	0.00 s
Transformations	56/90	5.59 s	0.10 s

We addressed this task in three different ways:

1. We compared CSDPs with previously existing techniques for proving termination of CSR.
2. We compared the improvements introduced by the different CS processors which have been defined in this paper.
3. We participated in the CSR subcategory of the 2007 International Termination Competition.

In the following subsections, we provide more details about this experimental evaluation.

### 13.1. CSDPs vs. other techniques for proving termination of CSR

Several methods have been developed to prove termination of CSR for a given CS-TRS  $(\mathcal{R}, \mu)$ . Two main approaches have been investigated so far:

1. *Direct proofs*, which are based on using  $\mu$ -reduction orderings (see [63]) such as the (context-sensitive) recursive path orderings [12] and polynomial orderings [26,48,49]. These are orderings  $>$  on terms that can be used to directly compare the left- and right-hand sides of the rules in order to conclude the  $\mu$ -termination of the TRS.
2. *Indirect proofs*, which obtain a proof of the  $\mu$ -termination of  $\mathcal{R}$  as a proof of termination of a transformed TRS  $\mathcal{R}_\Theta^\mu$  (where  $\Theta$  represents the transformation). If we are able to prove *termination* of  $\mathcal{R}_\Theta^\mu$  (using the standard methods), then the  $\mu$ -termination of  $\mathcal{R}$  is ensured.

We used MU-TERM to compare all these techniques with respect to the aforementioned benchmark examples. The results of this comparison are summarized in Table 1.

**Remark 15.** A number of transformations  $\Theta$  from TRSs  $\mathcal{R}$  and replacement maps  $\mu$  that produce TRSs  $\mathcal{R}_\Theta^\mu$  have been investigated by Lucas (transformation L [43]), Zantema (transformation Z [63]), Ferreira and Ribeiro (transformation FR [22]), and Giesl and Middeldorp (transformations<sup>9</sup> GM, sGM, and C [30,31]), see [31,50] for recent surveys about these transformations which also include a thorough analysis about their relative power. All these transformations were considered in our experiments, so the item “Transformations” in Table 1 concentrates the *joint* impact of all of them.

From the benchmarks summarized in Table 1, we clearly conclude that the CSDP-framework is the most powerful technique for proving termination of CSR. Actually, all the examples that were solved by using CSRPO or polynomial orderings were also solved using CSDPs. With regard to transformations, there is only one example (namely, Ex9\_Luc06, which can be solved by using transformation GM) that could not be solved with our current implementation.

**Example 30.** The following nonterminating TRS  $\mathcal{R}$  can be used to compute the list of prime numbers by using the well-known Erathostenes sieve<sup>10</sup> [30]:

```

primes → sieve(from(s(0)))
from(x) → cons(x, from(s(x)))
head(cons(x, y)) → x
sieve(cons(x, y)) → cons(x, filt(x, sieve(y)))
tail(cons(x, y)) → y
if(true, x, y) → x
if(false, x, y) → y
filt(s(s(x)), cons(y, z)) → if(div(s(s(x)), y), filt(s(s(x)), z), cons(y, filt(s(s(x)), z)))

```

<sup>9</sup> The labels for these transformations correspond to the ones introduced in [50].

<sup>10</sup> Without appropriate rules for defining symbol `div`, the TRS has no complete computational meaning. However, we take it here as given in [30] for the purpose of comparing different techniques for proving termination of CSR by transformation.

**Table 2**  
Comparison among CS processors.

Tool version	Narrowing	Non- $\mu$ -replacing projection	Subterm	Proved	Total time	Average time
1	No	No	No	54/90	3.00 s	0.05 s
2	No	No	Yes	62/90	0.55 s	0.01 s
3	No	Yes	No	57/90	0.82 s	0.01 s
4	No	Yes	Yes	65/90	0.49 s	0.01 s
5	Yes	No	No	54/90	3.22 s	0.06 s
6	Yes	No	Yes	62/90	2.64 s	0.04 s
7	Yes	Yes	No	57/90	1.27 s	0.02 s
8	Yes	Yes	Yes	65/90	0.31 s	0.00 s

Consider the replacement map  $\mu$  for the signature  $\mathcal{F}$  given by:

$$\mu(\text{cons}) = \mu(\text{if}) = \{1\} \text{ and } \mu(f) = \{1, \dots, ar(f)\} \text{ for all } f \in \mathcal{F} - \{\text{cons}, \text{if}\}.$$

From the termination point of view, this example is interesting because, since its introduction in Giesl and Middeldorp's paper [30], no automatic proof of termination has been reported. In sharp contrast, termination of CSR for this TRS and replacement map  $\mu$  is easily proved by using the techniques developed in this paper. In particular, the context-sensitive dependency graph contains no cycle.

### 13.2. Contribution of the different CS processors

In our implementation of the CSDP-framework, besides processor  $\text{Proc}_{\text{SCC}}$ , the subterm processors in Section 11 and the  $\mu$ -reduction-pair CS processors in Section 10 are the most frequently used (in this order). The impact of the CS processors in Sections 11 and 12 is summarized in Table 2. Our benchmarks show that the CS processors described in Section 11 play an important role in our proofs. The subterm processors  $\text{Proc}_{\text{SubColl}}$  and  $\text{Proc}_{\text{SubColl}}$  are quite efficient, but the ones that are based on simple projections for non- $\mu$ -replacing arguments ( $\text{Proc}_{\text{NRP}}$  and  $\text{Proc}_{\text{NRP2}}$ ) also increase the power and the speed of the CSDP technique. Furthermore, these two groups of CS processors are complementary: the extra problems that are specifically solved by them are different. Narrowing is useful for simplifying the graph, but it does not play an important role in the benchmarks because it is only applied to solve two examples (which can be solved without narrowing as well). Furthermore, it must be used carefully because recomputing the graph can be expensive in that case. Complete details of our experiments can be found here:

<http://zenon.dsic.upv.es/muterm/benchmarks/csdp/>.

### 13.3. CSDPs at the 2007 International Termination Competition

In 2007, AProVE [32] was the only tool (besides MU-TERM) implementing specific methods for proving termination of CSR. Both AProVE and MU-TERM participated in the CSR subcategory of the 2007 International Termination Competition. AProVE participated with a termination expert for CSR which, given a CS-TRS  $(\mathcal{R}, \mu)$ , successively tries different transformations  $\Theta$  for proving termination of CSR (which are enumerated in Remark 15, i.e.,  $\Theta \in \{C, FR, GM, L, sGM, Z\}$ ). It then uses a huge variety of different and complementary techniques to prove termination of rewriting (according to the DP-framework) on the obtained TRS  $\mathcal{R}_{\Theta}^{\mu}$ . Actually, AProVE is currently the most powerful tool for proving termination of TRSs and implements most existing results and techniques regarding DPs and related techniques.

However, MU-TERM's implementation of CSDPs was able to beat AProVE in the CSR category (MU-TERM was able to prove 68 of the 90 examples; AProVE proved 64), thus demonstrating that CSDPs are actually a very powerful technique for proving termination of CSR.

## 14. Related work

The first presentation of the context-sensitive dependency pairs was given in [3]. This paper is an extended and revised version of [3, 4]. We provide complete proofs for all results,<sup>11</sup> and also present many examples about the use of the theory. The main conceptual differences between [3, 4] and this paper are the following:

1. In this paper, we have investigated two different notions of minimal non- $\mu$ -terminating terms: the so-called *strongly minimal terms* ( $\mathcal{T}_{\infty, \mu}$ , which are introduced in this paper) and the *minimal terms* ( $\mathcal{M}_{\infty, \mu}$ ), which were introduced in [3] and further investigated in [4]. The combined use of these notions leads to a better development of the theory. This has brought new essential results, remarkably Theorem 1, which is the basis (at the level of pure context-sensitive rewriting) of the new notions of CSDP and minimal chain.

<sup>11</sup> We report and fix some bugs in previous papers.

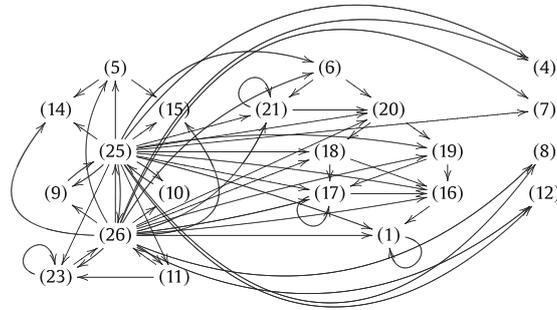


Fig. 6. Context-sensitive dependency graph of Example 1 following [3].

2. Although most of the ideas in this first part of the paper (Section 3) were present in [4, Section 3], we make some aspects explicit that were only implicit there. For instance, the essential notion of *hidden term* (a consequence of Lemma 5 which is further developed in Lemma 6 and Proposition 4) was implicit in [4, Section 3], but only the notion of *hidden symbol* was made explicit. Actually, the proofs of the aforementioned results in this paper correspond (with minor changes) to those of Lemmas 3.4 and 3.5, and Proposition 3.6 in [4], respectively.
3. The notion of context-sensitive dependency pairs was first introduced in [3, Definition 1], but the narrowing condition that we have now included for the noncollapsing CSDPs is new. This condition is inspired in the recent extension of the DP-method to Order-Sorted TRSs [53]. In this paper, we have elaborated it in depth to show that it is actually a natural requirement (see Section 3.4). In [53], it has already been shown that including 'narrowability' in the usual definition of dependency pair can be useful to automatically prove termination of rewriting. Similar considerations are valid for CSR.
4. In [3], a notion of minimal chain was introduced but not really used in the main results. Actually, the notion of minimal chain in this paper is completely different from the old one and is a consequence of the analysis of infinite  $\mu$ -rewrite sequences developed in Section 3. Furthermore, in this paper, the notion of minimal chain of pairs is essential for the definition of the context-sensitive dependency graph and the development of the CSDP-framework in Section 7.
5. The notion of context-sensitive dependency graph was first introduced in [3] and further refined in [4] thanks to the introduction of the hidden symbols. The definition in this paper introduces a new refinement through the notion of 'narrowable hidden term' and shows a nice symmetry between the arcs associated to noncollapsing and collapsing pairs. Furthermore, the new definition leads to a great simplification of the computed graph: for the CS-TRS in Example 1, compare the graph in Fig. 6 (corresponding to [3]) with the new graph in Fig. 5.
6. The estimation of the CSDG in [3, 4] was an adaptation of the one by Arts and Giesl [10] to the context-sensitive setting. In this paper, we have defined a new estimation of the CSDG on the basis of the most recent proposal by Giesl et al. [34].
7. The definition of a CSDP-framework for the mechanization of proofs of termination of CSR using CSDPs is new. A number of processors introduced here had a kind of counterpart in [3] (for instance, the use of  $\mu$ -reduction orderings was formalized in [3, Theorem 4] and the subterm criterion for noncollapsing pairs was formalized in [3, Theorem 5]) or in [4] (for instance, the narrowing transformation in [4, Theorem 5.3]), but they were not formulated as processors.
8. This paper introduces a number of new processors that can be used for proving termination of CSR: the SCC processor,<sup>12</sup> the processors for filtering or transforming collapsing pairs (see Section 9), the use of argument filterings,<sup>13</sup> the use of the subterm criterion with collapsing pairs (Theorem 13), etc.
9. Finally, for the first time, we have considered how to *disprove* termination of CSR within the CSDP framework (processor  $\text{Proc}_{\text{Chf}}$  in Theorem 5).

#### 14.1. CSDPs vs. DPs and a piece of history

The first attempt to develop a theory of dependency pairs for CSR started more than 10 years ago when the third author of this paper asked Thomas Arts (who was preparing the first presentation of the dependency pair method [9]) about the possibility of extending the dependency pair approach to CSR. Arts immediately noticed that the main problem of extending the existing results for ordinary rewriting to CSR was the possibility of having variables that are not replacing in the left-hand sides of the rules but that become replacing in the corresponding right-hand side. This is what we now call *migrating variables*. After this first failed attempt, the focus moved to transformations of CS-TRSs  $(\mathcal{R}, \mu)$  into ordinary TRSs  $\mathcal{R}_\Theta^\mu$  (where  $\Theta$  represents the transformation) in such a way that termination of  $\mathcal{R}_\Theta^\mu$  implies the  $\mu$ -termination of  $\mathcal{R}$  [31, 50].

<sup>12</sup> This is mentioned in [3, Section 4.2] but without any formal description.

<sup>13</sup> This was briefly mentioned at the end of [3, Section 4.2] but was never formalized.

During the spring of 2006, MU-TERM was being revised in preparation for its participation in the 2006 International Termination Competition, which was organized by Claude Marché. The idea of adapting DPs to CSR came up again. A first correct version of context-sensitive dependency pairs that did *not* at the time consider collapsing pairs was the following:

**Definition 15** (First preliminary version of CSDPs). Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let

$$\begin{aligned} \text{DP}_1(\mathcal{R}, \mu) = & \left\{ l^\sharp \rightarrow s^\sharp \mid l \rightarrow r \in R, r \triangleright_\mu s, \text{root}(s) \in \mathcal{D}, l \not\prec_\mu s \right\} \\ & \cup \left\{ l^\sharp \rightarrow \text{MUSUBTERM}(x) \mid l \rightarrow r \in R, x \in \text{Var}^\mu(r) - \text{Var}^\mu(l) \right\} \\ & \cup \left\{ \text{MUSUBTERM}(f(x_1, \dots, x_k)) \rightarrow \text{MUSUBTERM}(x_i) \mid f \in \mathcal{F}, i \in \mu(f) \right\} \\ & \cup \left\{ \text{MUSUBTERM}(f(x_1, \dots, x_k)) \rightarrow f^\sharp(x_1, \dots, x_k) \mid f \in \mathcal{D} \right\} \end{aligned}$$

with  $\mu^\sharp(f) = \mu(f)$  if  $f \in \mathcal{F}$ ,  $\mu^\sharp(f^\sharp) = \mu(f)$  if  $f \in \mathcal{D}$ , and  $\mu^\sharp(\text{MUSUBTERM}) = \emptyset$ .

We handle migrating variables  $x$  by enclosing them inside a term  $\text{MUSUBTERM}(x)$  which (after instantiating  $x$  by means of a substitution  $\sigma$ ) would be able to start the search for a  $\mu$ -replacing subterm  $s = f(s_1, \dots, s_k)$  which (after marking its root symbol  $f$  as  $f^\sharp$ ) is able to connect with the left-hand side of the next CSDP in a sequence. The notion of *chain* of CSDPs that was used here was essentially the standard one. All pairs were treated in the very same way and the only difference was that pairs were connected by using CSR instead of ordinary rewriting.

The implementation of the CSDPs in Definition 15 did not work very well in practice. The structure of pairs which dealt with migrating variables introduced many arcs in the corresponding graph and, therefore, many cycles. Thus, the following proposal was considered instead.

**Definition 16** (Second preliminary version of CSDPs). Let  $\mathcal{R} = (\mathcal{F}, R) = (\mathcal{C} \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $\text{DP}_2(\mathcal{R}, \mu) = \text{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu)$  where:

$$\begin{aligned} \text{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu) = & \left\{ l^\sharp \rightarrow s^\sharp \mid l \rightarrow r \in R, r \triangleright_\mu s, \text{root}(s) \in \mathcal{D}, l \not\prec_\mu s \right\} \\ \text{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu) = & \left\{ l^\sharp \rightarrow U_{l,f,x}(x) \mid l \rightarrow r \in R, f \in \mathcal{D}, x \in \text{Var}^\mu(r) - \text{Var}^\mu(l) \right\} \\ & \cup \left\{ U_{l,f,x}(f(x_1, \dots, x_k)) \rightarrow f^\sharp(x_1, \dots, x_k) \mid l \rightarrow r \in R, f \in \mathcal{D}, x \in \text{Var}^\mu(r) - \text{Var}^\mu(l) \right\} \end{aligned}$$

and  $\mu^\sharp(f) = \mu(f)$  if  $f \in \mathcal{F}$ ,  $\mu^\sharp(f^\sharp) = \mu(f)$  if  $f \in \mathcal{D}$ , and  $\mu^\sharp(U_{l,f,x}) = \{1\}$  for all rules  $l \rightarrow r$ , symbols  $f$ , and variables  $x$  originating one of these symbols.

Here, migrating variables  $x$  are enclosed inside a term  $U_{l,f,x}(x)$  which (after instantiating  $x$  by means of a substitution  $\sigma$ ) would be able to connect any  $\mu$ -replacing subterm  $s = f(s_1, \dots, s_k)$  (with  $f$  a defined symbol) with the left-hand side of the next CSDP in a sequence. Note that no explicit  $\mu$ -replacing subterm search is possible with this new definition of CSDP. Instead, this requirement was moved to the definition of chain. Now, although these dependency pairs still remain as the ‘traditional ones’, a clear distinction was made between two kinds of CSDPs: those that were obtained from the nonvariable parts of the right-hand sides of the rules ( $\text{DP}_{2,\mathcal{F}}(\mathcal{R}, \mu)$  in Definition 16) and those that were introduced to treat the migrating variables ( $\text{DP}_{2,\mathcal{X}}(\mathcal{R}, \mu)$  in Definition 16). Both kinds of CSDPs were clearly distinguished in the new definition of chain and the  $\mu$ -subterm requirement was used to describe how chains of such CSDPs are built.

A version of MU-TERM that implemented the CSDPs in Definition 16 was submitted for participation in the *Context-Sensitive* (sub)category of the 2006 International Termination Competition (June 2006). We are grateful to Claude Marché for providing a copy of the folder where the MU-TERM outcome was stored. It is now available at the following URL:

<http://zenon.dsic.upv.es/muterm/benchmarks/ic10/muterm-2006/benchmarks.html>

A further evolution led to the definition of CSDP which was finally published in [3]. In sharp contrast to the standard dependency pair approach, where all dependency pairs have tuple symbols  $f^\sharp$  both in the left- and right-hand sides, we finally took the definitive step to also consider *collapsing* pairs having a single *variable* in the right-hand side as the most elegant, concise and expressive way to reflect the effect of the migrating variables in the termination behavior of CSR. This is one of the most important and original contributions of the paper.

### 15. CSDPs vs. noncollapsing CSDPs

In [1], a transformation of collapsing pairs into ‘ordinary’ (i.e., noncollapsing) pairs is introduced. The transformation uses the following notion.

**Definition 17** (Hiding context [1, Definition 7]). Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . The function symbol  $f$  *hides the argument*  $i$  if there is a rule  $l \rightarrow r \in \mathcal{R}$  with  $r \triangleright_{\mu} f(r_1, \dots, r_i, \dots, r_n)$ ,  $i \in \mu(f)$ , and  $r_i$  contains a defined symbol or a variable at an active position. A context  $C$  is *hiding* iff  $C = \square$  or  $C$  has the form  $f(t_1, \dots, t_{i-1}, C', t_{i+1}, \dots, t_n)$  where  $f$  hides the argument  $i$  and  $C'$  is a hiding context.

The notion of CSDPs that is given in [1] is the following:

**Definition 18** [1, Definition 9]. Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . If  $DP_{\mathcal{X}}(\mathcal{R}, \mu) \neq \emptyset$ , we introduce a fresh *unhiding tuple symbol*  $\cup$  and the following *unhiding DPs*:

- $s \rightarrow \cup(x)$  for every  $s \rightarrow x \in DP_{\mathcal{X}}(\mathcal{R}, \mu)$ ,
- $\cup(f(x_1, \dots, x_i, \dots, x_n)) \rightarrow \cup(x_i)$  for every function symbol  $f$  of any arity  $n$  and every  $1 \leq i \leq n$  where  $f$  hides position  $i$ ,
- $\cup(t) \rightarrow t^{\sharp}$  for every hidden term  $t$ .

Let  $DP_u(\mathcal{R}, \mu)$  be the set of all unhiding DPs (where  $DP_u(\mathcal{R}, \mu) = \emptyset$  whenever  $DP_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$ ). Then  $DP'(\mathcal{R}, \mu) = DP_{\mathcal{F}}(\mathcal{R}, \mu) \cup DP_u(\mathcal{R}, \mu)$ .

The corresponding definition of *chain* is, essentially, the standard one [10], but  $\mu$ -rewriting (with  $\mathcal{R}$ ) is used for connecting pairs.

**Definition 19** [1, Definition 11]. Let  $\mathcal{P}$  and  $\mathcal{R}$  be TRSs and let  $\mu$  be a replacement map. We extend  $\mu$  to tuple symbols by defining  $\mu(f^{\sharp}) = \mu(f)$  for all  $f \in \mathcal{D}$  and  $\mu(\cup) = \emptyset$ . A sequence of pairs  $u_1 \rightarrow v_1, u_2 \rightarrow v_2, \dots$  from  $\mathcal{P}$  is a  $(\mathcal{P}, \mathcal{R}, \mu)$ -*chain* if there is a substitution  $\sigma$  with  $\sigma(v_i) \xrightarrow{\ast}_{\mathcal{R}, \mu} \sigma(u_{i+1})$  and  $\sigma(v_i)$  is  $(\mathcal{R}, \mu)$ -terminating for all  $i$ .

Using these definitions, a characterization of termination of CSR is given.

**Theorem 17** [1, Theorem 12]. A TRS  $\mathcal{R}$  is  $\mu$ -terminating if and only if there is no infinite  $(DP'(\mathcal{R}, \mu), \mathcal{R}, \mu)$ -chain.

On the basis of these definitions and results, Alarcón et al. [1, Section 4] develop a CSDP framework.

### 15.1. Comparing CSDPs and noncollapsing CSDPs

As discussed in Section 14.1, the idea of providing a definition of CSDPs that does not use collapsing pairs cannot be considered as the main contribution of [1]; in 2006 there was an implementation of CSDPs without collapsing pairs (namely the one which corresponds to Definition 16). Actually, Definition 18 is very close to Definition 15 (i.e., the first correct notion of CSDP developed in 2006) if we write  $\cup$  instead of  $MUSUBTERM$  in Definition 15. The crucial differences between Definition 15 and Definition 18 are the use of *hiding contexts* (Definition 17) and the use of *hidden terms* (Definition 3). As discussed in [1, Section 3], the notion of hiding context is a refinement of the notion of hidden term described in this paper (and previously approached in [4]), see Section 3.2.

Indeed, the notion of hiding context is the most important contribution of [1] from the theoretical side. The notion of hiding context can be easily integrated in the CSDP framework discussed in this paper. This has been carried out in [27, 28, 37], where an extension of our CSDP framework was developed to appropriately integrate this notion. Within this new approach, Definition 18 could be incorporated to the CSDP framework by using the following modified version of Theorem 8, which defines the appropriate CS processor.

**Theorem 18.** Let  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$  be TRSs and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $u \rightarrow x \in \mathcal{P}_{\mathcal{X}}$  and

$$P_u = \{u \rightarrow U(x)\} \\ \cup \{U(f(x_1, \dots, x_k)) \rightarrow U(x_i) \mid f \in \mathcal{F}, 1 \leq i \leq \text{ar}(f) \text{ and } f \text{ hides } i\} \\ \cup \{U(t) \rightarrow t^{\sharp} \mid t \in \mathcal{NHT}_{\mathcal{P}}\}$$

where  $U$  is a fresh symbol. Let  $P' = (\mathcal{G} \cup \{U\}, P')$ , where  $P' = (P - \{u \rightarrow x\}) \cup P_u$ , and  $\mu'$  which extends  $\mu$  by  $\mu'(U) = \emptyset$ . Then, the processor  $\text{Proc}_{\text{Ctx}}$  given by

$$\text{Proc}_{\text{Ctx}}(\mathcal{P}, \mathcal{R}, \mu) = \{(P', \mathcal{R}, \mu')\}$$

is sound and complete.

The proof of this result would be analogous to the one for Theorem 8 with the proviso, in our definition of chain of pairs (Definition 5), that the contexts  $C_i[\ ]_p$  which are used for handling collapsing pairs are now *hiding contexts*. In contrast to

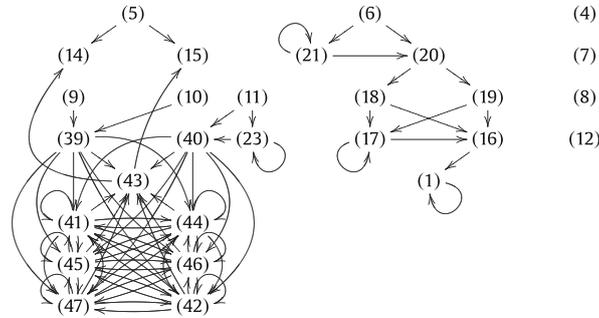


Fig. 7. Context-sensitive dependency graph of Example 1 according to [1].

processor  $\text{Proc}_{\text{eColl}}$  in Theorem 8,  $\text{Proc}_{\text{hCtx}}$  has the advantage of introducing fewer rules due to the use of the notion of *hiding* in Definition 17. Obviously, this could lead to simpler proofs when it is used.

In this paper, we have shown that collapsing pairs are an essential part of the theoretical description of termination of CSR. Actually, Definition 18 explicitly uses them to introduce the new unhiding pairs. This shows that the most basic notion when modeling the termination behavior of CSR is that of collapsing pair and that unhiding pairs should be better considered as an ingredient for handling collapsing pairs in proofs of termination (as implemented by processor  $\text{Proc}_{\text{hCtx}}$  above).

#### 15.2. Use of CSDPs and noncollapsing CSDPs

The application of Definition 18 at the very beginning of the termination analysis of CS-TRSs (as done in [1]) often leads to obtaining a more complex dependency graph. For instance, we would replace the collapsing CSDPs (25) and (26) by the following ones:

$$\text{TAIL}(\text{cons}(x, xs)) \rightarrow \text{U}(xs) \quad (39)$$

$$\text{TAKE}(s(n), \text{cons}(x, xs)) \rightarrow \text{U}(xs) \quad (40)$$

$$\text{U}(\text{incr}(x)) \rightarrow \text{U}(x) \quad (41)$$

$$\text{U}(\text{incr}(\text{oddNs})) \rightarrow \text{INCR}(\text{oddNs}) \quad (42)$$

$$\text{U}(\text{oddNs}) \rightarrow \text{ODDNS} \quad (43)$$

$$\text{U}(\text{rep2}(x)) \rightarrow \text{U}(x) \quad (44)$$

$$\text{U}(\text{zip}(x, y)) \rightarrow \text{U}(x) \quad (45)$$

$$\text{U}(\text{zip}(x, y)) \rightarrow \text{U}(y) \quad (46)$$

$$\text{U}(\text{cons}(x, y)) \rightarrow \text{U}(x) \quad (47)$$

to obtain the graph in Fig. 7, which should be compared with the CSDG for the same example in Fig. 5. On the other hand, if  $\mathcal{P}$  contains no collapsing pairs (as happens if Definition 18 is used to compute the dependency pairs of a CS-TRS), then Definition 19 is subsumed by our notion of chain of pairs (Definition 5). This means that, after using processor  $\text{Proc}_{\text{eColl}}$  in Theorem 8 to remove collapsing pairs in the component  $\mathcal{P}$  of a CS problem  $(\mathcal{P}, \mathcal{R}, \mu)$ , we could use all CS processors developed in [1], some of which have not been discussed in our paper (for instance, the *instantiation processor* [1, Theorem 24]). Also, the CS processors that are developed here can be used in any implementation following [1].

**Remark 16.** Note that, although the definition of chain in [1] (see Definition 19) is apparently closer to the standard one [35, Definition 3], this does *not* mean that we can use or easily 'translate' existing DP-processors (see [35]) to be used with CSR.

The narrowing processor provides a striking example. Example 29 shows that the application of the narrowing processor to the TRSs  $\mathcal{P}$  and  $\mathcal{R}$  in the example is *not* correct due to the lack of *strong  $\mu$ -conservativeness* of the  $\mu$ -narrowed pair in  $\mathcal{P}$ . Since  $\mathcal{P}$  has no collapsing pair, one could think (following a naïve interpretation of [1]) that the narrowing processor of the DP-framework (see [35, Theorem 31]), which does not take into account the replacement restrictions, should work with CSR without difficulties, which is not the case.

**Table 3**  
Summary of processors used in MU-TERM-IC.

Processors	Applied in
SCC processor (Section 8)	94/94 problems
Processors based on subterm criteria (Section 11)	56/94 problems
Processors based on reduction pairs (Section 10)	50/94 problems
Basic processors (Section 7)	28/94 problems
Narrowing processor (Section 12)	3/94 problems

Thus, a CSDP framework that is based on Definitions 18 and 19 *does not boil down to the DP-framework*, and a careful consideration of the replacement restrictions is necessary before being able to use any DP-processor with CSR.

### 15.3. Experimental evaluation

We have performed an experimental evaluation of the use of CSDPs vs. the ones in Definition 18 as follows: we prepared two versions of MU-TERM: MU-TERM-LPAR08 and MU-TERM-IC. The tool MU-TERM-LPAR08 first applies Theorem 18 to remove all collapsing pairs (as one would do when working within the approach described in [1]) and then uses the CS processors described in both this paper and in [1] to achieve termination proofs. On the other hand, MU-TERM-IC implements the CSDP framework that we have described here (with the modifications developed in [27,28,37]).

On a collection of 109 examples, both tools succeeded on the very same ones (94 proofs of termination). However, MU-TERM-IC performed globally faster. Furthermore, we *did not need to use*  $\text{Proc}_{\text{Ctx}}$  in the proofs with MU-TERM-IC. This suggests that (in contrast to what we claimed in [1] when the integration of the notion of hiding context into the CSDP framework was pending), collapsing pairs do not represent any drawback for automatically proving termination of CSR. Detailed benchmarks are at the following URL: <http://zenon.dsic.upv.es/muterm/benchmarks/ic10/muterm-2009/benchmarks.html> Table 3 shows the use of the different processors in these benchmarks. The interpretation of the frequency of use for the different processors should take into account the following *strategy* for invoking them in MU-TERM-IC when CS problems are treated: first, we try the basic (infinite and finite) processors. If some of them succeed, we are done; otherwise, we continue as follows:

1. SCC processor.
2. Subterm criterion processors.
3. Reduction pair (RP) processors with polynomial and matrix interpretations over the reals [6,7,49,51].
4. Narrowing processor.

Interestingly, *all* processors are used at least once during the proofs.

## 16. Conclusions

We have analyzed the structure of infinite context-sensitive rewrite sequences starting from minimal non- $\mu$ -terminating terms (Theorem 1). This knowledge is used to provide an appropriate definition of context-sensitive dependency pair (Definition 4), and the related notion of chain (Definition 5). In sharp contrast to the standard dependency pair approach, where all dependency pairs have tuple symbols  $f^\sharp$  in both the left- and right-hand sides, we have *collapsing* dependency pairs that have a single *variable* in the right-hand side. These variables reflect the effect of the *migrating* variables on the termination behavior of CSR. At the level of *minimal chains*, however, the contrast with the ordinary DP approach is somehow recovered by a nice symmetry arising from the central notion of *hidden term* (Definition 3): a noncollapsing pair  $u \rightarrow v$  is followed by a pair  $u' \rightarrow v'$  if  $\sigma(v)$   $\mu$ -rewrites into  $\sigma(u')$  for some substitution  $\sigma$ ; a collapsing pair  $u \rightarrow v$  is followed by a pair  $u' \rightarrow v'$  if there is a *hidden term*  $t$  such that  $\sigma(t)^\sharp$   $\mu$ -rewrites into  $\sigma(u')$  for some substitution  $\sigma$ . We have shown how to use the context-sensitive dependency pairs in proofs of termination of CSR. As in Arts and Giesl's approach, the absence of infinite minimal chains of dependency pairs from  $\text{DP}(\mathcal{R}, \mu)$  characterizes the  $\mu$ -termination of  $\mathcal{R}$  (Theorems 2 and 3).

We have provided a suitable adaptation of the *dependency pair framework* to CSR by defining appropriate notions of *CS problem* (Definition 6) and *CS processor* (Definition 7). We have described a number of sound and (most of them) complete CS processors that can be used in any practical implementation of the CSDP-framework. In particular, we have introduced the notion of (estimated) *context-sensitive (dependency) graph* (Definitions 8 and 10) and the associated CS processor (Theorem 6). We have also described some CS processors for removing or transforming collapsing pairs from CS problems (Theorems 7 and 8). We are also able to use  $\mu$ -reduction pairs (Definition 11) and argument filterings to ensure the absence of infinite chains of pairs (Theorems 9, 10, and 11). We have adapted Hirokawa and Middeldorp's *subterm criterion* which permits concluding the absence of infinite minimal chains by paying attention only to the pairs in the corresponding CS problem (Theorems 12 and 13). Following this appealing idea, we have also introduced two new processors that work in a similar way but use a very basic kind of ordering instead of the subterm relation (Theorems 14 and 15). Narrowing context-sensitive dependency pairs have also been investigated. It is helpful to simplify or restructure the dependency graph and eventually simplify the proof of termination (Theorem 16).

We have implemented these ideas as part of the termination tool MU-TERM [2,47]. The implementation and practical use of the developed techniques yield a novel and powerful framework that improves the current state-of-the-art of methods for proving termination of CSR. Actually, CSDPs were an essential ingredient for MU-TERM in winning the context-sensitive subcategory of the 2007 competition of termination tools.

For future work, we plan to extend the basic CSDP-framework described in this paper with further CS processors integrating the *usable rules* for CSR [29] and proofs of termination of *innermost* CSR using CSDPs [5].

### Acknowledgments

We thank Jürgen Giesl and his group of the RWTH Aachen (especially Fabian Emmes, Carsten Fuhs, Peter Schneider-Kamp, and René Thiemann) for many fruitful discussions about CSDPs. We also thank the anonymous referees for many useful remarks.

### Appendix A

#### A.1. Proofs of Section 3

**Lemma 2.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is not  $\mu$ -terminating, then there is a subterm  $t$  of  $s$  ( $s \triangleright t$ ) such that  $t \in \mathcal{T}_{\infty, \mu}$ .

**Proof.** By structural induction. If  $s$  is a constant symbol, it is obvious: take  $t = s$ . If  $s = f(s_1, \dots, s_k)$ , then we proceed by contradiction. If there is no subterm  $t$  of  $s$  such that  $t \in \mathcal{T}_{\infty, \mu}$ , then  $s \notin \mathcal{T}_{\infty, \mu}$ . Since  $s$  is not  $\mu$ -terminating, there is a strict subterm  $t$  of  $s$  ( $s \triangleright t$ ) that is not  $\mu$ -terminating. By the Induction Hypothesis, there is  $t' \in \mathcal{T}_{\infty, \mu}$  such that  $t \triangleright t'$ . Then, we have  $s \triangleright t'$ , thus leading to a contradiction.  $\square$

**Lemma 3.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ , and  $s \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $s$  is not  $\mu$ -terminating, then there is a  $\mu$ -replacing subterm  $t$  of  $s$  such that  $t \in \mathcal{M}_{\infty, \mu}$ .

**Proof.** By structural induction. If  $s$  is a constant symbol, it is obvious: take  $t = s$ . If  $s = f(s_1, \dots, s_k)$ , then we proceed by contradiction. If there is no  $\mu$ -replacing subterm  $t$  of  $s$  such that  $t \in \mathcal{M}_{\infty, \mu}$ , then  $s \notin \mathcal{M}_{\infty, \mu}$ , i.e., there is a strict  $\mu$ -replacing subterm  $t$  of  $s$  which is not  $\mu$ -terminating. We have  $t = s|_p$  for some  $p \in \text{Pos}^{\mu}(s) - \{\Lambda\}$ . By the Induction Hypothesis,  $t$  contains a  $\mu$ -replacing subterm  $t'$  which belongs to  $\mathcal{M}_{\infty, \mu}$ , i.e.,  $t' = t|_q$  for some  $q \in \text{Pos}^{\mu}(t)$ . By Proposition 1,  $p \cdot q \in \text{Pos}^{\mu}(s)$ . Thus,  $t'$  is a  $\mu$ -replacing subterm of  $s$  that belongs to  $\mathcal{M}_{\infty, \mu}$ , thus leading to a contradiction.  $\square$

**Lemma 4.** Let  $\mathcal{R}$  be a TRS,  $\mu \in M_{\mathcal{R}}$ , and  $t \in \mathcal{M}_{\infty, \mu}$ . If  $t \xrightarrow{> \Lambda}^* u$  and  $u$  is non- $\mu$ -terminating, then  $u \in \mathcal{M}_{\infty, \mu}$ .

**Proof.** All  $\mu$ -rewritings below of  $t$  the root are issued on  $\mu$ -replacing and  $\mu$ -terminating terms that remain  $\mu$ -terminating by Lemma 1. Then, all strict  $\mu$ -replacing subterms of  $u$  (which are the ones that can be originated or transformed by  $\mu$ -rewritings from  $t$  to  $u$ ) are  $\mu$ -terminating. Since  $u$  is non- $\mu$ -terminating,  $u \in \mathcal{M}_{\infty, \mu}$ .  $\square$

#### 16.1. Proofs of Section 3.2

**Lemma 5.** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  and  $\sigma$  be a substitution. If there is a rule  $l \rightarrow r \in R$  such that  $\sigma(l) \not\triangleright_{\mu} t$  and  $\sigma(r) \triangleright_{\mu} t$ , then there is no  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$ . Furthermore, there is a term  $t' \in \mathcal{HT}$  such that  $r \triangleright_{\mu} t'$  and  $\sigma(t') = t$ .

**Proof.** By contradiction. If there is  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$ , then since variables in  $l$  are always below some function symbol we have  $\sigma(l) \triangleright t$ , leading to a contradiction.

Since there is no  $x \in \text{Var}(r)$  such that  $\sigma(x) \triangleright t$  but we have that  $\sigma(r) \triangleright_{\mu} t$ , then there is a nonvariable and non- $\mu$ -replacing position  $p \in \text{Pos}_{\mathcal{F}}(r) - \text{Pos}^{\mu}(r)$  of  $r$ , such that  $\sigma(r|_p) = t$ . Then, we let  $t' = r|_p$ . Note that  $t' \in \mathcal{HT}$ .  $\square$

**Lemma 6.** Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Let  $A$  be a  $\mu$ -rewrite sequence  $t_1 \leftrightarrow t_2 \leftrightarrow \dots \leftrightarrow t_n$  with  $t_i \in \mathcal{M}_{\infty, \mu}$  for all  $i$ ,  $1 \leq i \leq n$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_1 \not\triangleright_{\mu} t$  and  $t_n \triangleright_{\mu} t$ , then  $t = \sigma(s)$  for some  $s \in \mathcal{DHT}$  and substitution  $\sigma$ .

**Proof.** By induction on  $n$ :

1. If  $n = 1$ , then it is vacuously true because  $t_1 \not\triangleright_{\mu} t$  and  $t_1 \triangleright_{\mu} t$  do not simultaneously hold.

2. If  $n > 1$ , then we assume that  $t_1 \not\triangleright_{\mu} t$  and  $t_n \triangleright_{\mu} t$ . We consider two cases:
- If  $t_{n-1} \triangleright_{\mu} t$ , then by the induction hypothesis the conclusion follows.
  - If  $t_{n-1} \not\triangleright_{\mu} t$  does not hold, then, since assuming  $t_{n-1} \triangleright_{\mu} t$  leads to a contradiction (because  $t_{n-1} \in \mathcal{M}_{\infty, \mu}$  in the hypothesis implies that  $t \notin \mathcal{M}_{\infty, \mu}$ ), we have that  $t_{n-1} \not\triangleright_{\mu} t$ . Let  $l \rightarrow r \in R$  be such that  $t_{n-1} = C[\sigma(l)]$  and  $t_n = C[\sigma(r)]$  for some context  $C[\ ]$  and substitution  $\sigma$ . Then, in particular,  $\sigma(l) \not\triangleright_{\mu} t$  and, since  $t_n \triangleright_{\mu} t$  there must be  $\sigma(r) \triangleright_{\mu} t$ . Thus, by Lemma 5 we conclude that  $t = \sigma(s)$  for some  $s \in \mathcal{HT}$  and substitution  $\sigma$ . Since  $t \in \mathcal{M}_{\infty, \mu}$ , it follows that  $\text{root}(t) = \text{root}(s) \in \mathcal{D}$ . Thus,  $s \in \mathcal{DHT}$ .  $\square$

**Proposition 4.** Let  $R$  be a TRS and  $\mu \in M_R$ . Consider a finite or infinite sequence of the form  $t_1 \xrightarrow{\Delta} s_1 \triangleright_{\mu} t'_2 \xrightarrow{\Delta} t_2 \xrightarrow{\Delta} s_2 \triangleright_{\mu} t'_3 \xrightarrow{\Delta} t_3 \cdots$  with  $t_j, t'_j \in \mathcal{M}_{\infty, \mu}$  for all  $j \geq 1$ . If there is a term  $t \in \mathcal{M}_{\infty, \mu}$  such that  $t_i \triangleright_{\mu} t$  for some  $i \geq 1$ , then  $t_1 \triangleright_{\mu} t$  or  $t = \sigma(s)$  for some  $s \in \mathcal{DHT}$  and substitution  $\sigma$ .

**Proof.** By induction on  $i$ :

- If  $i = 1$ , it is trivial.
- If  $i > 1$  and  $t_i \triangleright_{\mu} t$ , then we consider two cases:
  - If  $t_{i-1} \triangleright_{\mu} t$ , then by the induction hypothesis, the conclusion follows.
  - If  $t_{i-1} \not\triangleright_{\mu} t$  does not hold, then let  $l \rightarrow r \in R$  and  $\sigma$  be such that  $t_{i-1} = \sigma(l)$  and  $s_{i-1} = \sigma(r) \triangleright_{\mu} t'_i$ . Since  $t_{i-1} \not\triangleright_{\mu} t$  leads to a contradiction (because  $t_{i-1} \in \mathcal{M}_{\infty, \mu}$  implies that  $t \notin \mathcal{M}_{\infty, \mu}$ ), we have that  $t_{i-1} \not\triangleright_{\mu} t$ . We consider two cases:
    - If  $t'_i \triangleright_{\mu} t$ , then, since  $t'_i, t \in \mathcal{M}_{\infty, \mu}$ , the case  $t'_i \triangleright_{\mu} t$  is excluded and the only possibility is that  $t'_i \triangleright_{\mu} t$ . Then, since  $\sigma(l) = t_{i-1} \not\triangleright_{\mu} t$  and  $\sigma(r) \triangleright_{\mu} t'_i \triangleright_{\mu} t$ , i.e.,  $\sigma(r) \triangleright_{\mu} t$ , by Lemma 5 we conclude that  $t = \sigma(s)$  for some  $s \in \mathcal{HT}$  and substitution  $\sigma$ . Since  $t \in \mathcal{M}_{\infty, \mu}$ , it follows that  $\text{root}(t) = \text{root}(s) \in \mathcal{D}$ . Thus,  $s \in \mathcal{DHT}$ .
    - If  $t'_i \not\triangleright_{\mu} t$ , then, by applying Lemma 4 and Lemma 6 to the  $\mu$ -rewrite sequence  $t'_i \xrightarrow{\Delta}_{R, \mu} t_i$ , the conclusion follows.  $\square$

## References

- B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, R. Thiemann, Improving context-sensitive dependency pairs, in: I. Cervesato, H. Veith, A. Voronkov (Eds.), Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08, LNAI, vol. 5330, Springer, Berlin, 2008, pp. 636–651.
- B. Alarcón, R. Gutiérrez, J. Iborra, S. Lucas, Proving termination of context-sensitive rewriting with MU-TERM, Electronic Notes in Theoretical Computer Science 188 (2007) 105–115.
- B. Alarcón, R. Gutiérrez, S. Lucas, Context-sensitive dependency pairs, in: S. Arun-Kumar, N. Garg (Eds.), Proceedings of the XXVI Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06, LNCS, vol. 4337, Springer, Berlin, 2006, pp. 297–308.
- B. Alarcón, R. Gutiérrez, S. Lucas, Improving the context-sensitive dependency graph, Electronic Notes in Theoretical Computer Science 188 (2007) 91–103.
- B. Alarcón, S. Lucas, Termination of innermost context-sensitive rewriting using dependency pairs, in: B. Konev, F. Wolter (Eds.), Proceedings of the Sixth International Symposium on Frontiers of Combining Systems, FroCoS'07, LNAI, vol. 4720, Springer, Berlin, 2007, pp. 73–87.
- B. Alarcón, S. Lucas, R. Navarro-Marset, Proving termination with matrix interpretations over the reals, in: A. Geser, J. Waldmann (Eds.), Proceedings of the 10th International Workshop on Termination, WST'09, June 2009, pp. 12–15.
- B. Alarcón, S. Lucas, R. Navarro-Marset, Using matrix interpretations over the reals in proofs of termination, in: F. Lucio, G. Moreno, R. Peña (Eds.), Proceedings of the Ninth Spanish Conference on Programming and Computer Languages, PROLE'09, Universidad del País Vasco, 2009, pp. 255–264.
- M. Alpuente, S. Escobar, B. Gramlich, S. Lucas, On-demand strategy annotations revisited: an improved on-demand evaluation strategy, Theoretical Computer Science 411 (2) (2010) 504–541.
- T. Arts, Automatically proving termination and innermost normalisation of term rewriting systems, Ph.D. Thesis, Universiteit Utrecht, 1997.
- T. Arts, J. Giesl, Termination of term rewriting using dependency pairs, Theoretical Computer Science 236 (1–2) (2000) 133–178.
- F. Baader, T. Nipkow, Term Rewriting and All That, Cambridge University Press, Cambridge, 1998.
- C. Borralleras, S. Lucas, A. Rubio, Recursive path orderings can be context-sensitive, in: A. Voronkov (Ed.), Proceedings of the XXVIII Conference on Automated Deduction, CADE'02, LNAI, vol. 2392, Springer, Berlin, 2002, pp. 314–331.
- R. Bruni, J. Meseguer, Semantic foundations for generalized rewrite theories, Theoretical Computer Science 351 (1) (2006) 386–414.
- M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Marti-Oliet, J. Meseguer, C. Talcott, All about Maude – a high-performance logical framework, in: Lecture Notes in Computer Science, vol. 4350, Springer, Berlin, 2007.
- N. Dershowitz, Termination by abstraction, in: B. Demoen, V. Lifschitz (Eds.), Proceedings of the 20th International Conference on Logic Programming, ICLP'04, LNCS, vol. 3132, Springer, Berlin, 2004, pp. 1–18.
- F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving termination of membership equational programs, Proceedings of the 2004 ACM SIGPLAN Symposium on Partial Evaluation and Program Manipulation, PEPM'04, ACM Press, New York, 2004, pp. 147–158.
- F. Durán, S. Lucas, C. Marché, J. Meseguer, X. Urbain, Proving operational termination of membership equational programs, Higher-Order and Symbolic Computation 21 (1–2) (2008) 59–88.
- J. Endrullis, Jambox: Automated termination proofs for string and term rewriting. Available from: <<http://joerg.endrullis.de/jambox.html>>.
- J. Endrullis, D. Hendriks, From outermost to context-sensitive rewriting, in: R. Treinen (Ed.), Proceedings of the 20th International Conference on Rewriting Techniques and Applications, RTA'09, LNCS, vol. 5595, Springer, Berlin, 2009, pp. 305–319.
- J. Endrullis, J. Waldmann, H. Zantema, Matrix interpretations for proving termination of term rewriting, Journal of Automated Reasoning 40 (2–3) (2008) 195–220.
- M.-L. Fernández, Relaxing monotonicity for innermost termination, Information Processing Letters 93 (2005) 117–123.
- M.C.F. Ferreira, A.L. Ribeiro, Context-sensitive AC-rewriting, in: P. Narendran, M. Rusinowitch (Eds.), Proceedings of the 10th International Conference on Rewriting Techniques and Applications, RTA'99, LNCS, vol. 1631, Springer, Berlin, 1999, pp. 286–300.

- [23] K. Futatsugi, J. Goguen, J.-P. Jouannaud, J. Meseguer, Principles of OBJ2, in: Conference Record of the 12th Annual ACM Symposium on Principles of Programming Languages, POPL'85, ACM Press, New York, 1985, pp. 52–66.
- [24] K. Futatsugi, A. Nakagawa, an overview of CAFE specification environment – an algebraic approach for creating, verifying, and maintaining formal specification over networks, in: Proceedings of the First International Conference on Formal Engineering Methods, 1997.
- [25] J. Giesl, T. Arts, E. Ohlebusch, Modular termination proofs for rewriting using dependency pairs, *Journal of Symbolic Computation* 34 (1) (2002) 21–58.
- [26] B. Gramlich, S. Lucas, Simple termination of context-sensitive rewriting, in: B. Fischer, E. Visser (Eds.), Proceedings of the III ACM SIGPLAN Workshop on Rule-Based Programming, RULE'02, ACM Press, New York, 2002, pp. 29–41.
- [27] R. Gutiérrez, S. Lucas, Mechanizing proofs of termination in the context-sensitive dependency pairs framework, in: F. Lucio, G. Moreno, R. Peña (Eds.), Proceedings of the Ninth Spanish Conference on Programming and Computer Languages, PROLE'09, Universidad del País Vasco, 2009, pp. 265–274.
- [28] R. Gutiérrez, S. Lucas, Mechanizing proofs of termination with context-sensitive dependency pairs, in: A. Geser, J. Waldmann (Eds.), Proceedings of the 10th International Workshop on Termination, WST'09, HTWK Leipzig, 2009, pp. 43–46.
- [29] R. Gutiérrez, S. Lucas, X. Urbain, Usable rules for context-sensitive rewrite systems, in: A. Voronkov (Ed.), Proceedings of the 19th International Conference on Rewriting Techniques and Applications, RTA'08, LNCS, vol. 5117, Springer, Berlin, 2008, pp. 126–141.
- [30] J. Giesl, A. Middeldorp, Transforming context-sensitive rewrite systems, in: P. Narendran, M. Rusinowitch (Eds.), Proceedings of the X International Conference on Rewriting Techniques and Applications, RTA'99, LNCS, vol. 1631, Springer, Berlin, 1999, pp. 271–285.
- [31] J. Giesl, A. Middeldorp, Transformation techniques for context-sensitive rewrite systems, *Journal of Functional Programming* 14 (4) (2004) 379–427.
- [32] J. Giesl, P. Schneider-Kamp, R. Thiemann, AProVE 1.2: automatic termination proofs in the dependency pair framework, in: U. Furbach, N. Shankar (Eds.), Proceedings of the Third International Conference on Automated Reasoning, IJCAR'06, LNAI, vol. 4130, Springer, Berlin, 2006, pp. 281–286. Available from: <<http://aprove.informatik.rwth-aachen.de/>>.
- [33] J. Giesl, R. Thiemann, P. Schneider-Kamp, The dependency pair framework: combining techniques for automated termination proofs, in: F. Baader, A. Voronkov (Eds.), Proceedings of the XI International Conference on Logic for Programming Artificial Intelligence and Reasoning, LPAR'04, LNAI, vol. 3452, Springer, Berlin, 2004, pp. 301–331.
- [34] J. Giesl, R. Thiemann, P. Schneider-Kamp, Proving and disproving termination of higher-order functions, in: B. Gramlich (Ed.), Proceedings of the Fifth International Workshop on Frontiers of Combining Systems, FroCoS'05, LNAI, vol. 3717, Springer, Berlin, 2005, pp. 216–231.
- [35] J. Giesl, R. Thiemann, P. Schneider-Kamp, S. Falke, Mechanizing and improving dependency pairs, *Journal of Automatic Reasoning* 37 (3) (2006) 155–203.
- [36] J.A. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, J.-P. Jouannaud, Introducing OBJ, in: J. Goguen, G. Malcolm (Eds.), *Software Engineering with OBJ: Algebraic Specification in Action*, Kluwer, Dordrecht, 2000, pp. 1–2.
- [37] R. Gutiérrez, Context-sensitive dependency pairs framework, Master's thesis, Departamento de Sistemas Informáticos y Computación, Universitat Politècnica de València, València, Spain, December 2008.
- [38] N. Hirokawa, A. Middeldorp, Dependency pairs revisited, in: V. van Oostrom (Ed.), Proceedings of the XV International Conference on Rewriting Techniques and Applications, RTA'04, LNCS, vol. 3091, Springer, Berlin, 2004, pp. 249–268.
- [39] N. Hirokawa, A. Middeldorp, Automating the dependency pair method, *Information and Computation* 199 (2005) 172–199.
- [40] N. Hirokawa, A. Middeldorp, Tyrolean termination tool: techniques and features, *Information and Computation* 205 (2007) 474–511.
- [41] P. Hudak, S.J. Peyton-Jones, P. Wadler, Report on the functional programming language Haskell: a non-strict, purely functional language, *Sigplan Notices* 27 (5) (1992) 1–164.
- [42] K. Kusakari, M. Nakamura, Y. Toyama, Argument filtering transformation, in: G. Nadathur (Ed.), Proceedings of the International Conference on Principles and Practice of Declarative Programming, PPDP'99, LNCS, vol. 1702, 1999, pp. 47–61.
- [43] S. Lucas, Termination of context-sensitive rewriting by rewriting, in: F. Meyer auf der Heide, B. Monien (Eds.), Proceedings of the 23rd International Colloquium on Automata, Languages and Programming, ICALP'96, LNCS, vol. 1099, Springer, Berlin, 1996, pp. 122–133.
- [44] S. Lucas, Context-sensitive computations in functional and functional logic programs, *Journal of Functional and Logic Programming* 1998 (1) (1998) 1–61.
- [45] S. Lucas, Termination of on-demand rewriting and termination of OBJ programs, in: Proceedings of the Third International Conference on Principles and Practice of Declarative Programming, PPDP'01, ACM Press, New York, 2001, pp. 82–93.
- [46] S. Lucas, Context-sensitive rewriting strategies, *Information and Computation* 178 (1) (2002) 293–343.
- [47] S. Lucas, MU-TERM: a tool for proving termination of context-sensitive rewriting, in: V. van Oostrom (Ed.), Proceedings of the XV International Conference on Rewriting Techniques and Applications, RTA'04, LNCS, vol. 3091, Springer, Berlin, 2004, pp. 200–209. Available from: <<http://zenon.dsic.upv.es/muterm/>>.
- [48] S. Lucas, Polynomials for proving termination of context-sensitive rewriting, in: I. Walukiewicz (Ed.), Proceedings of the VII International Conference on Foundations of Software Science and Computation Structures, FOSSACS'04, LNCS, vol. 2987, Springer, Berlin, 2004, pp. 318–332.
- [49] S. Lucas, Polynomials over the reals in proofs of termination: from theory to practice, *RAIRO Theoretical Informatics and Applications* 39 (3) (2005) 547–586.
- [50] S. Lucas, Proving Termination of Context-Sensitive Rewriting by Transformation, *Information and Computation* 204 (12) (2006) 1782–1846.
- [51] S. Lucas, Practical use of polynomials over the reals in proofs of termination, in: Proceedings of the Ninth International Symposium on Principles and Practice of Declarative Programming, PPDP'07, ACM Press, New York, 2007, pp. 39–50.
- [52] S. Lucas, J. Meseguer, Operational termination of membership equational programs: the order-sorted way, *Electronic Notes in Theoretical Computer Science* 238 (3) (2009) 207–225.
- [53] S. Lucas, J. Meseguer, Order-sorted dependency pairs, in: Proceedings of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08, ACM Press, New York, 2008, pp. 108–119.
- [54] J. McCarthy, Recursive functions of symbolic expressions and their computations by machine. Part I, *Communications of the ACM* 3 (4) (1960) 184–195.
- [55] A. Middeldorp, Approximating dependency graphs using tree automata techniques, in: R. Goré, A. Leitsch, T. Nipkow (Eds.), Proceedings of the International Joint Conference on Automated Reasoning, IJCAR'01, LNAI, vol. 2083, 2001, pp. 593–610.
- [56] A. Middeldorp, Approximations for strategies and termination, *Electronic Notes in Computer Science* 70 (6) (2002).
- [57] E.G.J.M.H. Nöcker, J.E.W. Smetsers, M.C.J.D. van Eekelen, M.J. Plasmeijer, Concurrent clean, in: E.H.L. Aarts, J. Leeuwen, M. Rem (Eds.), Proceedings of the Parallel Architectures and Languages Europe, PARLE'91, LNCS, vol. 506, Springer, Berlin, 1992, pp. 202–219.
- [58] E. Ohlebusch, *Advanced Topics in Term Rewriting*, Springer, Berlin, 2002.
- [59] F. Schernhammer, B. Gramlich, Termination of lazy rewriting revisited, *Electronic Notes in Theoretical Computer Science* 204 (2008) 35–51.
- [60] F. Schernhammer, B. Gramlich, VMTL-A modular termination laboratory, in: R. Treinen (Ed.), Proceedings of the 20th International Conference on Rewriting Techniques and Applications, RTA'09, LNCS, vol. 5595, Springer, Berlin, 2009, pp. 285–294.
- [61] TeReSe, *Term Rewriting Systems*, Cambridge University Press, Cambridge, 2003.
- [62] R. Thiemann, The DP framework for proving termination of term rewriting, Ph.D. Thesis. Available as Technical Report AIB-2007-17, RWTH Aachen, Germany, 2007.
- [63] H. Zantema, Termination of context-sensitive rewriting, in: H. Comon (Ed.), Proceedings of the VII International Conference on Rewriting Techniques and Applications, RTA'97, LNCS, vol. 1232, Springer, Berlin, 1997, pp. 172–186.

## 8.12 Proving Termination in the Context-Sensitive Dependency Pair Framework

7. R. Gutiérrez and S. Lucas. **Proving Termination in the Context-Sensitive Dependency Pairs Framework.** In P. Ölveczky, editor, *Proc. of 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*, volume 6381 of *Lecture Notes in Computer Science*, pages 19–35. Springer-Verlag, 2010.

## Proving Termination in the Context-Sensitive Dependency Pair Framework\*

Raúl Gutiérrez<sup>1</sup> and Salvador Lucas<sup>1</sup>

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*  $\mu$  satisfying  $\mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$  for every function symbol  $f$  of arity  $\text{ar}(f)$  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.

2 Raúl Gutiérrez and Salvador Lucas

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

$$\begin{array}{ll}
 \text{gt}(0, y) \rightarrow \text{false} & \text{p}(0) \rightarrow 0 \\
 \text{gt}(s(x), 0) \rightarrow \text{true} & \text{p}(s(x)) \rightarrow x \\
 \text{gt}(s(x), s(y)) \rightarrow \text{gt}(x, y) & \text{minus}(x, y) \rightarrow \text{if}(\text{gt}(y, 0), \text{minus}(\text{p}(x), \text{p}(y)), x) \\
 \text{if}(\text{true}, x, y) \rightarrow x & \text{div}(0, s(y)) \rightarrow 0 \\
 \text{if}(\text{false}, x, y) \rightarrow y & \text{div}(s(x), s(y)) \rightarrow s(\text{div}(\text{minus}(x, y), s(y)))
 \end{array}$$

with  $\mu(\text{if}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{ar}(f)\}$  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:

$$\text{minus}(0, 0) \rightarrow_{\mathcal{R}}^* \text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0) \rightarrow_{\mathcal{R}}^* \dots \text{minus}(0, 0), \dots \rightarrow_{\mathcal{R}}^* \dots$$

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

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  $\ell \rightarrow r$  such that  $r$  contains some *defined* symbol<sup>1</sup>  $\mathbf{g}$ . Intuitively, we can think of these rules as representing possible (direct or indirect) recursive calls. Recursion paths associated to each rule  $\ell \rightarrow r$  are represented as new rules  $u \rightarrow v$  (called *dependency pairs*) where  $u = \mathbf{f}^\sharp(\ell_1, \dots, \ell_k)$  if  $\ell = \mathbf{f}(\ell_1, \dots, \ell_k)$  and  $v = \mathbf{g}^\sharp(s_1, \dots, s_m)$  if  $s = \mathbf{g}(s_1, \dots, s_m)$  is a subterm of  $r$  and  $\mathbf{g}$  is a defined symbol. The notation  $\mathbf{f}^\sharp$  for a given symbol  $\mathbf{f}$  means that  $\mathbf{f}$  is *marked*. In practice, we often capitalize  $\mathbf{f}$  and use  $\mathbf{F}$  instead of  $\mathbf{f}^\sharp$  in our examples. For this reason, the dependency pair technique starts by considering a new TRS  $\text{DP}(\mathcal{R})$  which contains all these dependency pairs for each  $\ell \rightarrow r \in \mathcal{R}$ . The rules in  $\mathcal{R}$  and  $\text{DP}(\mathcal{R})$  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  $\ell \rightarrow r$  which are considered to build the *context-sensitive dependency pairs*  $\ell^\sharp \rightarrow s^\sharp$  must be  $\mu$ -replacing terms. In sharp contrast with the dependency pair approach, though, we also need *collapsing dependency pairs*  $u \rightarrow x$  where  $u$  is obtained from the left-hand side  $\ell$  of a rule  $\ell \rightarrow r$  in the usual way, i.e.,  $u = \ell^\sharp$  but  $x$  is a *migrating variable* which is  $\mu$ -replacing in  $r$  but which only occurs at *non- $\mu$ -replacing positions* in  $\ell$  [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

<sup>1</sup> A symbol  $\mathbf{g} \in \mathcal{F}$  is defined in  $\mathcal{R}$  if there is a rule  $\ell \rightarrow r$  in  $\mathcal{R}$  whose left-hand side  $\ell$  is of the form  $\mathbf{g}(\ell_1, \dots, \ell_k)$  for some  $k \geq 0$ .

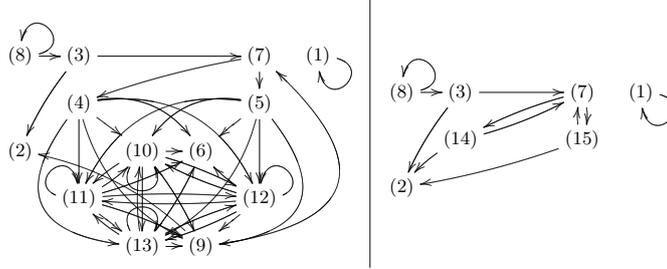


Fig. 1. Dependency graph for Example 1 following [6] (left) and [9] (right)

the non- $\mu$ -replacing parts of the terms involved in the infinite sequence until a *migrating* variable within a rule  $\ell \rightarrow r$  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 lose 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):*

$$\begin{array}{ll}
 \text{GT}(s(x), s(y)) \rightarrow \text{GT}(x, y) & (1) \\
 \text{M}(x, y) \rightarrow \text{GT}(y, 0) & (2) \\
 \text{D}(s(x), s(y)) \rightarrow \text{M}(x, y) & (3) \\
 \text{IF}(\text{true}, x, y) \rightarrow \text{U}(x) & (4) \\
 \text{IF}(\text{false}, x, y) \rightarrow \text{U}(y) & (5) \\
 \text{U}(p(x)) \rightarrow \text{P}(x) & (6) \\
 \text{M}(x, y) \rightarrow \text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x) & (7) \\
 \text{D}(s(x), s(y)) \rightarrow \text{D}(\text{minus}(x, y), s(y)) & (8) \\
 \text{U}(\text{minus}(p(x), p(y))) \rightarrow \text{M}(p(x), p(y)) & (9) \\
 \text{U}(p(x)) \rightarrow \text{U}(x) & (10) \\
 \text{U}(p(y)) \rightarrow \text{U}(y) & (11) \\
 \text{U}(\text{minus}(x, y)) \rightarrow \text{U}(x) & (12) \\
 \text{U}(\text{minus}(x, y)) \rightarrow \text{U}(y) & (13)
 \end{array}$$

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:

$$\begin{array}{ll}
 \text{GT}(s(x), s(y)) \rightarrow \text{GT}(x, y) & (1) \\
 \text{M}(x, y) \rightarrow \text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x) & (7) \\
 \text{M}(x, y) \rightarrow \text{GT}(y, 0) & (2) \\
 \text{D}(s(x), s(y)) \rightarrow \text{D}(\text{minus}(x, y), s(y)) & (8) \\
 \text{D}(s(x), s(y)) \rightarrow \text{M}(x, y) & (3) \\
 \text{IF}(\text{true}, x, y) \rightarrow x & (14) \\
 \text{IF}(\text{false}, x, y) \rightarrow y & (15)
 \end{array}$$

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  $\mu$ -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  $p, q, \dots$  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  $\Lambda$ . The set of positions of a term  $t$  is  $\mathcal{P}os(t)$ . Positions of nonvariable symbols  $f \in \mathcal{F}$  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  $t \geq s$  if  $s = t|_p$  for some  $p \in \mathcal{P}os(t)$  and  $t \triangleright s$  if  $t \geq s$  and  $t \neq s$ . The symbol labeling the root of  $t$  is denoted as  $\text{root}(t)$ . A *substitution* is a mapping  $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$  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  $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{X})$  with a ‘hole’  $\square$  (a fresh constant symbol). A *rewrite rule* is an ordered pair  $(\ell, r)$ , written  $\ell \rightarrow r$ , with  $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $\ell \notin \mathcal{X}$  and  $\text{Var}(r) \subseteq \text{Var}(\ell)$ . The left-hand side (*lhs*) of the rule is  $\ell$  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  $c \in \mathcal{C}$ , called *constructors* and symbols  $f \in \mathcal{D}$ , called *defined symbols*, where  $\mathcal{D} = \{\text{root}(\ell) \mid \ell \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ .

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

$t \triangleright_{\mu} s$  to denote that  $s$  is a *non- $\mu$ -replacing strict subterm* of  $t$ , i.e., there is a *non- $\mu$ -replacing position*  $p \in \mathcal{P}os(t) \setminus \mathcal{P}os^{\mu}(t)$  such that  $s = t|_p$ . In CSR, we (only) contract  *$\mu$ -replacing redexes*:  $t$   $\mu$ -rewrites to  $s$ , written  $t \hookrightarrow_{\mathcal{R},\mu} s$  (or  $t \xrightarrow{p}_{\mathcal{R},\mu} s$  to make position  $p$  explicit), iff there are  $\ell \rightarrow r \in R$ ,  $p \in \mathcal{P}os^{\mu}(t)$  and a substitution  $\sigma$  such that  $t|_p = \sigma(\ell)$  and  $s = t[\sigma(r)]_p$ ;  $t \xrightarrow{q}_{\mathcal{R},\mu} s$  means that the  $\mu$ -rewrite step is applied below position  $q$ , i.e.,  $p > q$ . We say that a variable  $x$  is *migrating* in  $\ell \rightarrow r \in R$  if  $x \in \mathcal{V}ar^{\mu}(r) \setminus \mathcal{V}ar^{\mu}(\ell)$ . A term  $t$  is  *$\mu$ -terminating* if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \dots \hookrightarrow_{\mathcal{R},\mu} t_n \hookrightarrow_{\mathcal{R},\mu} \dots$  starting from  $t$ . A TRS  $\mathcal{R} = (\mathcal{F}, R)$  is  *$\mu$ -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 $\mu$ -Rewrite Sequences

Let  $\mathcal{M}_{\infty,\mu}$  be a set of *minimal non- $\mu$ -terminating terms* in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty,\mu}$  if  $t$  is non- $\mu$ -terminating and every strict  *$\mu$ -replacing subterm*  $s$  of  $t$  (i.e.,  $t \triangleright_{\mu} s$ ) is  $\mu$ -terminating [7]. Minimal terms allow us to characterize infinite  $\mu$ -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- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{T}_{\infty,\mu}$  if  $t$  is non- $\mu$ -terminating and every strict subterm  $u$  (i.e.,  $t \triangleright u$ ) is  $\mu$ -terminating. It is obvious that  $\mathit{root}(t) \in \mathcal{D}$  for all  $t \in \mathcal{T}_{\infty,\mu}$ .*

Every non- $\mu$ -terminating term has a subterm that is strongly minimal. Then, given a non- $\mu$ -terminating term  $t$  we can always find a subterm  $t_0 \in \mathcal{T}_{\infty,\mu}$  of  $t$  which starts a *minimal infinite  $\mu$ -rewrite sequence* of the form  $t_0 \xrightarrow{\geq^A}_{\mathcal{R},\mu} \sigma_1(\ell_1) \xrightarrow{A}_{\mathcal{R},\mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq^A}_{\mathcal{R},\mu} \sigma_2(\ell_2) \xrightarrow{A}_{\mathcal{R},\mu} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{\geq^A}_{\mathcal{R},\mu} \dots$  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- $\mu$ -terminating terms  $t_i \in \mathcal{M}_{\infty,\mu}$  in the sequence are partially introduced by a  $\mu$ -replacing nonvariable subterm of the right-hand sides  $r_i$  of the rules  $\ell_i \rightarrow r_i$ .
- The minimal non- $\mu$ -terminating terms  $t_i \in \mathcal{M}_{\infty,\mu}$  in the sequence are introduced by instantiated *migrating variables*  $x_i$  of (the respective) rules  $\ell_i \rightarrow r_i$ , 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- $\mu$ -replacing positions in the right-hand sides of the rules (*hidden terms*) within a given (*hiding*) context.

We use the following functions [7, 9]:  $\mathit{REN}^{\mu}(t)$ , which *independently* renames all *occurrences* of  $\mu$ -replacing variables by using new fresh variables which are not

6 Raúl Gutiérrez and Salvador Lucas

in  $\mathcal{V}ar(t)$ , and  $\text{NARR}_{\mathcal{R}}^{\mu}(t)$ , which indicates whether  $t$  is  $\mu$ -narrowable<sup>2</sup> (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  $\ell \rightarrow r \in R$  such that  $t$  is a non- $\mu$ -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  $\mu$ -narrowable hidden terms headed by a defined symbol:

$$\mathcal{NHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D} \text{ and } \text{NARR}_{\mathcal{R}}^{\mu}(\text{REN}^{\mu}(t))\}$$

**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  $\ell \rightarrow r \in \mathcal{R}$  if  $r \triangleright_{\mu} f(r_1, \dots, r_n)$  for some terms  $r_1, \dots, r_n$ , and there is  $i \in \mu(f)$  such that  $r_i$  contains a  $\mu$ -replacing defined symbol (i.e.,  $\text{Pos}_{\mathcal{D}}^{\mu}(r_i) \neq \emptyset$ ) or a variable  $x \in (\text{Var}^{\mu}(\ell) \cap \text{Var}^{\mu}(r)) \setminus (\text{Var}^{\mu}(\ell) \cup \text{Var}^{\mu}(r))$  which is  $\mu$ -replacing in  $r_i$  (i.e.,  $x \in \text{Var}^{\mu}(r_i)$ ). A context  $C[\square]$  is hiding [6] if  $C[\square] = \square$ , or  $C[\square] = f(t_1, \dots, t_{i-1}, C'[\square], t_{i+1}, \dots, t_k)$ , where  $f$  hides position  $i$  and  $C'[\square]$  is a hiding context.*

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

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

These notions are used and combined to model infinite context-sensitive rewrite sequences starting from strongly minimal non- $\mu$ -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*

$$t = t_0 \xrightarrow{\geq_{\mathcal{R}, \mu}^A} \sigma_1(\ell_1) \xrightarrow{A} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq_{\mathcal{R}, \mu}^A} \sigma_2(\ell_2) \xrightarrow{A} \dots$$

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

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

## 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.

<sup>2</sup> A term  $s$   $\mu$ -narrows to the term  $t$  if there is a nonvariable position  $p \in \text{Pos}_{\mathcal{F}}^{\mu}(s)$  and a rule  $\ell \rightarrow r$  such that  $s|_p$  and  $\ell$  unify with mgu  $\sigma$ , 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  $\text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$  to be set of context-sensitive dependency pairs (CSDPs) where:

$$\begin{aligned} \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \triangleright_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\triangleright_\mu s, \text{NARR}_{\mathcal{R}}^\mu(\text{REN}^\mu(s))\} \\ \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow x \mid \ell \rightarrow r \in R, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell)\} \end{aligned}$$

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

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  $\mu$ -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  $\text{unh}(\mathcal{R}, \mu)$  as the TRS consisting of the following rules:

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

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

$$\begin{array}{ll} \text{minus}(p(x), p(y)) \rightarrow M(p(x), p(y)) & (16) \quad \text{minus}(x, y) \rightarrow y & (18) \\ p(x) \rightarrow P(x) & (17) \quad \text{minus}(x, y) \rightarrow x & (19) \end{array}$$

Definitions 3 and 4 lead to a suitable notion of *chain* which captures minimal infinite  $\mu$ -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  $s \rightarrow t \in \mathcal{S}$  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  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma$  satisfying that, for all  $i \geq 1$ ,

1. if  $v_i \notin \text{Var}(u_i) \setminus \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) = t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , and
2. if  $v_i \in \text{Var}(u_i) \setminus \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) \xrightarrow{\mathcal{S}_{\triangleright_\mu, \mu}}^* \circ \xrightarrow{\mathcal{A}}^* t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ .

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  $f(x_1, \dots, x_k) \rightarrow x_i$  for all  $f \in \mathcal{D}$  and  $i \in \mu(f)$  (where  $x_1, \dots, x_k$  are variables) are used in Item 1 of Definition 4, then Definition 5 yields the notion of chain in [9]; and if, additionally, rules  $f(x_1, \dots, x_k) \rightarrow f^\sharp(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$  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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ -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  $\tau$  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- $\mu$ -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  $\tau$ ,  $\tau$  is finite whenever  $\text{Proc}(\tau) \neq \text{no}$  and  $\forall \tau' \in \text{Proc}(\tau)$ ,  $\tau'$  is finite. A CS processor Proc is complete if for all CS problems  $\tau$ ,  $\tau$  is infinite whenever  $\text{Proc}(\tau) = \text{no}$  or  $\exists \tau' \in \text{Proc}(\tau)$  such that  $\tau'$  is infinite.

In order to prove the  $\mu$ -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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ . For every inner node labeled with  $\tau$ , there is a sound processor Proc satisfying one of the following conditions:

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

If all leaves of the tree are labeled with “yes”, then  $\mathcal{R}$  is  $\mu$ -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- $\mu$ -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:  $P_{\mathcal{X}} = \{u \rightarrow v \in P \mid v \in \mathcal{V}ar(u) \setminus \mathcal{V}ar^\mu(u)\}$  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  $P_{\cup}$  be given by the following rules:

- $u \rightarrow U(x)$  for every  $u \rightarrow x \in P_{\mathcal{X}}$ ,
- $U(s) \rightarrow U(t)$  for every  $s \rightarrow t \in \mathcal{S}_{>\mu}$ , and

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

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

Now, we can apply all CS processors from [6] and [9] which did not consider any  $S$  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- $\mu$ -replacing immediate subterms in the left-hand side is  $\mathcal{P}_{\mathcal{X}}^1 = \{f(u_1, \dots, u_k) \rightarrow x \in \mathcal{P}_{\mathcal{X}} \mid \exists i, 1 \leq i \leq k, i \notin \mu(f), x \in \text{Var}(u_i)\}$ .

**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  $s \rightarrow t \in \mathcal{S}_\sharp$ ,  $t \notin \mathcal{X}$  (2)  $\{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_\sharp\} \cap \mathcal{D} = \emptyset$  and (3) for all  $s \rightarrow t \in \mathcal{S}_\sharp$ , we have that  $s = f(s_1, \dots, s_k)$  and  $t = g(s_1, \dots, s_k)$  for some  $k \in \mathbb{N}$ , function symbols  $f, g \in \mathcal{H}$ , and terms  $s_1, \dots, s_k$ . Then, the processors  $\text{Proc}_{CollI}$  given by*

$$\text{Proc}_{CollI}(\tau) = \begin{cases} \emptyset & \text{if } \mathcal{P} = \mathcal{P}_{\mathcal{X}}^1 \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

*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  $\text{Proc}_{CollI}(\tau)$  to conclude that the CS problem  $\tau$  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  $\text{DP}(\mathcal{R})$  and there is an arc from a dependency pair  $u \rightarrow v$  to a dependency pair  $u' \rightarrow v'$  if there are substitutions  $\theta$  and  $\theta'$  such that  $\theta(v) \rightarrow_{\mathcal{R}}^* \theta'(u')$ . 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  $G(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  has  $\mathcal{P}$  as the set of nodes. Given  $u \rightarrow v, u' \rightarrow v' \in \mathcal{P}$ , there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$  if  $u \rightarrow v, u' \rightarrow v'$  is a minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain for some substitution  $\sigma$ .*

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.

10 Raúl Gutiérrez and Salvador Lucas

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

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

(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  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$ , which renames all subterms headed by a ‘defined’ symbol in  $\mathcal{R}$  by new fresh variables if it can be rewritten:

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

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

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

where  $y$  is a new fresh variable,  $[s]_i^f = \text{TCAP}_{\mathcal{R}}^{\mu}(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ . We assume that  $\ell$  shares no variable with  $f([t_1]_1^f, \dots, [t_k]_k^f)$  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  $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ ) has  $\mathcal{P}$  as the set of nodes and arcs which connect them as follows:*

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

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

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

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

where

$$- \mathcal{S}_{\mathcal{Q}} = \emptyset \text{ if } \mathcal{Q}_{\mathcal{X}} = \emptyset.$$

- $\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow t \mid s \rightarrow t \in \mathcal{S}_{\sharp}, \text{TCAP}_{\mathcal{R}}^{\mu}(t) \text{ and } u' \text{ unify for some } u' \rightarrow v' \in \mathcal{Q}\}$  if  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ .

is sound and complete.

*Example 6.* In Figure 1 (right) we show  $\text{EG}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$  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  $\text{Proc}_{\text{SCC}}$  to the initial CS problem  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$  for  $(\mathcal{R}, \mu)$  in Example 1, then we obtain the problems:  $(\mathcal{P}_1, \mathcal{R}, \emptyset, \mu^{\sharp})$ ,  $(\mathcal{P}_2, \mathcal{R}, \emptyset, \mu^{\sharp})$ ,  $(\mathcal{P}_3, \mathcal{R}, \mathcal{S}_3, \mu^{\sharp})$ , where  $\mathcal{S}_3 = \{(16), (18), (19)\}$ .

### 5.3 Reduction Triple Processor

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

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

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

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

<sup>3</sup> A binary relation  $R$  on terms is  $\mu$ -monotonic if for all terms  $s, t, t_1, \dots, t_k$ , and  $k$ -ary symbols  $f$ , whenever  $s R t$  and  $i \in \mu(f)$  we have  $f(t_1, \dots, t_{i-1}, s, \dots, t_k) R f(t_1, \dots, t_{i-1}, t, \dots, t_k)$ .

12 Raúl Gutiérrez and Salvador Lucas

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

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

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

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  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ . 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  $\text{Proc}_{RT}$  to the CS problem  $\tau$  by using the  $\mu$ -reduction triple  $(\succeq, >, \geq)$  where  $\geq$  and  $>$  are the orderings induced by the following polynomial interpretation (see [17] for missing notation and definitions):*

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

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

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

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

#### 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  $\pi$  is restricted to  $\mu$ -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:

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

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

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

**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)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_G \cup \mathcal{S}_\mu$  are noncollapsing, (3) for all  $s_i \rightarrow t_i \in \mathcal{S}_{\triangleright_\mu}$ ,  $\text{root}(s_i), \text{root}(t_i) \notin \text{Root}(\mathcal{P})$  and (4) for all  $s_i \rightarrow t_i \in \mathcal{S}_\mu$ ,  $\text{root}(s_i) \notin \text{Root}(\mathcal{P})$  and  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{P}_{\pi, \triangleright_\mu} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright_\mu \pi(v)\}$  and  $\mathcal{S}_{\pi, \triangleright_\mu} = \{s \rightarrow t \in \mathcal{S} \mid \pi(s) \triangleright_\mu \pi(t)\}$ . Then,  $\text{Proc}_{\text{subterm}}$  given by

$$\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright_\mu}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright_\mu}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \triangleright_\mu \\ & \text{and whenever } \mathcal{P}_X \neq \emptyset, \\ & \text{then } \pi(\mathcal{S}) \subseteq \triangleright_\mu \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

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  $\text{Proc}_{\text{subterm}}$  by using the projection  $\pi(\text{IF}) = 3$  and  $\pi(\text{M}) = 1$ :

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

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

## 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 NatILList .
  subsort NatList < NatILList .
  op 0 : -> Nat .
```

14 Raúl Gutiérrez and Salvador Lucas

```

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<sup>4</sup> (TPDB): TRS/CSR\_Maude/LengthOfFiniteLists\_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<sup>5</sup>.

## 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<sup>6</sup> 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

<sup>4</sup> <http://www.lri.fr/~marche/tpdb/>

<sup>5</sup> 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>

<sup>6</sup> 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
<b>AProVE</b>	34/37	3.084s
<b>Jambox</b>	28/37	2.292s
MU-TERM	34/37	1.277s
<b>VMTL</b>	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<sup>7</sup>, 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  $\text{Proc}_{eColl}$  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  $\text{Proc}_{eColl}$  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

<sup>7</sup> 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 lose 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 Innermost 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 Symposium 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>.

### 8.13 Proving Termination Properties with MU-TERM

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

## Proving Termination Properties with MU-TERM\*

Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas, and Rafael Navarro-Marset

ELP group, DSIC, Universidad Politécnica de Valencia  
Camino de Vera s/n, E-46022 Valencia, Spain

**Abstract.** MU-TERM is a tool which can be used to verify a number of termination properties of (variants of) Term Rewriting Systems (TRSs): termination of rewriting, termination of innermost rewriting, termination of order-sorted rewriting, termination of context-sensitive rewriting, termination of innermost context-sensitive rewriting and termination of rewriting modulo specific axioms. Such termination properties are essential to prove termination of programs in sophisticated rewriting-based programming languages. Specific methods have been developed and implemented in MU-TERM in order to efficiently deal with most of them. In this paper, we report on these new features of the tool.

### 1 Introduction

Handling typical programming language features such as sort/types and subtypes, evaluation modes (*eager/lazy*), programmable strategies for controlling the execution, rewriting modulo axioms and so on is outside the scope of many termination tools. However, such features can be very important to determine the termination behavior of programs. For instance, in Figure 1 we show a Maude [10] program encoding an *order-sorted* TRS which is terminating when the sorting information is taken into account but which is *nonterminating* as a TRS (i.e., disregarding sort information) [18]. The predicate `is-even` tests whether an integer number is even. When disregarding any information about sorts, the program `EVEN` is not terminating due to the last rule for `is-even`, which specifies a recursive call to `is-even`. However, when sorts are considered and the hierarchy among them is taken into account, such recursive call is not longer possible due to the need of binding variable `Y` of sort `NzNeg` to an expression `opposite(Y)` of sort `NzPos`, which is not possible in the (sub)sort hierarchy given by `EVEN`.

The notions coming from the already quite mature theory of termination of TRSs (orderings, reduction pairs, dependency pairs, semantic path orderings, etc.) provide a basic collection of abstractions for treating termination problems. For real programming languages, though, having appropriate adaptations, methods, and techniques for specific termination problems is essential. Giving support to *multiple* extensions of such *classical* termination notions is one of the main goals for developing a new version of our tool, MU-TERM 5.0:

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

\* Partially supported by EU (FEDER) and MICINN grant TIN 2007-68093-C02-02.

```

fmod EVEN is
  sorts Zero NzNeg Neg NzPos Pos Int Bool .
  subsorts Zero < Neg < Int .
  subsorts Zero < Pos < Int .
  op 0 : -> Zero .
  op s : Pos -> NzPos .
  op p : Neg -> NzNeg .
  ops true false : -> Bool .
  var X : Pos .
  eq opposite(p(0)) = s(0) .
  eq opposite(p(Y)) = s(opposite(Y)) .
  eq is-even(0) = true .
  eq is-even(s(0)) = false .
  eq is-even(s(s(X))) = is-even(X) .
  eq is-even(Y) = is-even(opposite(Y)) .
endfm
  subsorts NzNeg < Neg .
  subsorts NzPos < Pos .
  op is-even : Int -> Bool .
  op is-even : NzPos -> Bool .
  op is-even : NzNeg -> Bool .
  op opposite : NzNeg -> NzPos .
  var Y : NzNeg .

```

Fig. 1. Maude program

MU-TERM [23, 2] was originally designed to prove termination of *Context-Sensitive Rewriting* (CSR, [21]), where reductions are allowed only for specific arguments  $\mu(f) \subseteq \{1, \dots, k\}$  of the  $k$ -ary function symbols  $f$  in the TRS. In this paper we report on the new features included in MU-TERM 5.0, not only to improve its ability to prove termination of CSR but also to verify a number of *other* termination properties of (variants of) TRSs.

In contrast to *transformational* approaches which translate termination problems into a classical termination problem for TRSs, we have developed specific techniques to deal with termination of CSR, innermost CSR, order-sorted rewriting and rewriting modulo specific axioms (associative or commutative) by using dependency pairs (DPs, [7]). Our benchmarks show that direct methods lead to simpler, faster and more successful proofs. Moreover, MU-TERM 5.0 has been rewritten to embrace the dependency pair framework [17], a recent formulation of the dependency pair approach which is specially well-suited for mechanizing proofs of termination.

## 2 Structure and Functionality of MU-TERM 5.0

MU-TERM 5.0 consists of 47 Haskell modules with more than 19000 lines of code. A web-based interface and compiled versions in several platforms are available at the MU-TERM 5.0 web site. In the following, we describe its new functionalities.

### 2.1 Proving Termination of Context-Sensitive Rewriting

As in the unrestricted case [7], the *context-sensitive dependency pairs* (CSDPs, [3]) are intended to capture all possible function calls in infinite  $\mu$ -rewrite sequences. In [2], even though our quite ‘immature’ CSDP approach was one of

our major assets, MU-TERM still used *transformations* [15, 25] and the *context-sensitive recursive path ordering* (CSRPO, [9]) in many termination proofs. Since the developments in [2], many improvements and refinements have been made when dealing with termination proofs of CSR. The most important one has been the development of the *context-sensitive dependency pair framework* (CSDP framework, [3, 20]), for mechanizing proofs of termination of CSR. The central notion regarding termination proofs is that of *CS problem*; regarding mechanization of the proofs is that of *CS processor*. Most processors in the standard DP-framework [17] have been adapted to CSR and many specific ones have been developed (see [3, 20]). Furthermore, on the basis of the results in [28] we have implemented specific processors to prove the infiniteness of CS problems. Therefore, MU-TERM 5.0 is the first version of MU-TERM which is also able to disprove termination of CSR. In the following table, we compare the performance of MU-TERM 5.0 and the last reported version of the tool (MU-TERM 4.3 [2]) regarding its ability to prove termination of CSR over the context-sensitive category of the *Termination Problem Data Base*<sup>1</sup> (TPDB) which contains 109 examples<sup>2</sup>. The results show the power of the new CSDP framework in MU-TERM 5.0, not

Termination Tool	Total	Yes	No	CSDPs	CSRPO	Transf.	Average (sec)
MU-TERM <b>5.0</b>	99/109	95	4	99	0	0	0.95s
MU-TERM <b>4.3</b>	64/109	64	0	54	7	3	3.44s

**Table 1.** MU-TERM 4.3 compared to MU-TERM 5.0 in proving termination of CSR.

only by solving more examples in less time, but also disregarding the need of using transformations or CSRPO for solving them.

## 2.2 Proving Termination of Innermost CSR

Termination of *innermost* CSR (i.e., the variant of CSR where only the deepest  $\mu$ -replacing redexes are contracted) has been proved useful for proving termination of programs in *eager* programming languages like Maude and OBJ\* which permit to control the program execution by means of context-sensitive annotations. Techniques for proving termination of innermost CSR were first investigated in [14, 22]. In these papers, though, the original CS-TRS  $(\mathcal{R}, \mu)$  is *transformed* into a TRS whose *innermost* termination implies the innermost termination for  $(\mathcal{R}, \mu)$ . In [4], the *dependency pair method* [7] has been adapted to deal with termination proofs of innermost CSR. This is the first proposal of a *direct* method for proving termination of innermost CSR and MU-TERM was the first termination tool able to deal with it. Our experimental evaluation shows that the use of *innermost context-sensitive dependency pairs* (ICSDPs) highly improves over the performance of transformational methods for proving termination of innermost CSR: innermost termination of 95 of the 109 considered CS-TRSs could be

<sup>1</sup> See <http://termination-portal.org/wiki/TPDB>

<sup>2</sup> We have used version 7.0.2 of the TPDB.

proved by using ICSDPs; in contrast, only 60 of the 109 could be proved by using (a combination of) transformations and then using AProVE [16] for proving the innermost termination of the obtained TRS. Another important aspect of innermost CSR is its use for proving termination of CSR as part of the CSDP framework [1]. Under some conditions, termination of CSR and termination of innermost CSR coincide [14, 19]. We then switch from termination of CSR to termination of innermost CSR, for which we can apply the existing processors more successfully (see Section 2.6). Actually, we proceed like that in 30 – 50% of the CSR termination problems which are proved by MU-TERM 5.0 (depending on the particular benchmarks).

### 2.3 Proving Termination of Order-Sorted Rewriting

In *order-sorted rewriting*, sort information is taken into account to specify the kind of terms that function symbols can take as arguments. Recently, the *order-sorted dependency pairs* have been introduced and proved useful for proving termination of order-sorted TRSs [26]. As a remarkable difference w.r.t. the standard approach, we can mention the notion of *applicable rules* which are those rules which can eventually be used to rewrite terms of a given sort. Another important point is the use of order-sorted matching and unification. To our knowledge, MU-TERM 5.0 is the only tool which implements specific methods for proving termination of OS-TRSs<sup>3</sup>. Our benchmarks over the examples in the literature (there is no order-sorted category in the TPDB yet) show that the new techniques perform quite well. For instance, we can prove termination of the OS-TRS EVEN in Figure 1 automatically.

### 2.4 Proving Termination of $A \vee C$ -Rewriting

Recently, we have developed a suitable dependency pair framework for proving termination of  $A \vee C$ -rewrite theories [5]. An  $A \vee C$ -rewrite theory is a tuple  $\mathcal{R} = (\Sigma, E, R)$  where  $E$  is a set containing associative or commutative axioms associated to function symbols of the signature  $\Sigma$ . We have implemented the techniques described in [5] in MU-TERM. Even with only a few processors implemented, MU-TERM behaves well in the equational category of the TPDB, solving 39 examples out of 71. Obviously, we plan to investigate and implement more processors in this field. This is not the first attempt to prove termination of rewriting modulo axioms: CiME [11] is able to prove AC-termination of TRSs, and AProVE is able to deal with termination of rewriting modulo equations satisfying some restrictions.

### 2.5 Use of Rational Polynomials and Matrix Interpretations

Proofs of termination with MU-TERM 5.0 heavily rely on the generation of polynomial orderings using polynomial interpretations with rational coefficients [24].

<sup>3</sup> The Maude Termination Tool [12] implements a number of *transformations* from OS-TRSs into TRSs which can also be used for this purpose.

In this sense, recent improvements which are new with respect to the previous versions of MU-TERM reported in [2, 23] are the use of an autonomous SMT-based constraint-solver for rational numbers [8] and the use of matrix interpretations *over the reals* [6]. Our benchmarks show that polynomials over the rationals are used in around 25% of the examples where a polynomial interpretation is required during the successful proof. Matrix interpretations are used in less than 4% of the proofs.

## 2.6 Termination Expert

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

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

1. We check the system for extra variables (at active positions) in the right-hand side of the rules.
2. We check whether the system is innermost equivalent (see Section 2.2). If it is true, then we transform the problem into an innermost one.
3. Then, we obtain the corresponding dependency pairs, obtaining a (CS)DP problem. And now, recursively:
  - (a) Decision point between infinite processors and the *strongly connected component* (SCC) processor.
  - (b) Subterm criterion processor.
  - (c) Reduction triple (RT) processor with linear polynomials (LPoly) and coefficients in  $N_2 = \{0, 1, 2\}$ .
  - (d) RT processor with LPoly and coefficients in  $Q_2 = \{0, 1, 2, \frac{1}{2}\}$  and  $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{1}{4}\}$  (in this order).
  - (e) RT processor with simple mixed polynomials (SMPoly) and coefficients in  $N_2$ .
  - (f) RT processor with SMPoly and rational coefficients in  $Q_2$ .
  - (g) RT processor with 2-square matrices with entries in  $N_2$  and  $Q_2$ .
  - (h) Transformation processors (only twice to avoid nontermination of the strategy): instantiation, forward instantiation, and narrowing.
4. If the techniques above fail, then we use (CS)RPO.

The explanation of each processor can be found in [3, 20]. Note also that all processors are new with respect to MU-TERM 4.3 [2].

## 2.7 External use of MU-TERM

The Maude Termination Tool<sup>4</sup> (MTT [12]), which transforms proofs of termination of Maude programs into proofs of termination of CSR, use MU-TERM's expert as an external tool to obtain the proofs. The context-sensitive and order-sorted

<sup>4</sup> <http://www.lcc.uma.es/~duran/MTT>

features developed as part of MU-TERM 5.0 are essential to successfully handling *Maude* programs in MTT. The Knuth-Bendix completion tool `mkbTT` [29] is a modern completion tool that combines multi-completion with the use of termination tools. In the web version of the tool, the option to use MU-TERM as the external termination tool is available.

### 3 Conclusions

We have described MU-TERM 5.0, a new version of MU-TERM with new features for proving different termination properties like termination of *innermost* CSR, termination of *order-sorted rewriting* and termination of rewriting *modulo (associative or commutative) axioms*. Apart from that, a complete implementation of the *CSDP framework* [20] has been included in MU-TERM 5.0, leading to a much more powerful tool for proving termination of CSR. While transformations were used in MU-TERM 4.3, in MU-TERM 5.0 they are not used anymore. The research in the field has increased the number of examples which could be handled with CSDPs in 35 (see Table 1). Regarding proofs of *termination of rewriting*, from a collection of 1468 examples from the TPDB 7.0.2, MU-TERM 5.0 is able to prove (or disprove) termination of 835 of them. In contrast, MU-TERM 4.3 was able to deal with 503 only.

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

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

Thanks to the new developments reported in this paper, MU-TERM 5.0 has proven to be the most powerful tool for proving termination of CSR in the *context-sensitive* subcategory of the 2007, 2009, and 2010 editions of the International Competition of Termination Tools<sup>5</sup>. Moreover, in the *standard* subcategory, we have obtained quite good results in the 2009 and 2010 editions, being the third tool (among five) in solving more examples. We have also participated in the innermost category in the 2009 and 2010 editions and in the equational category in 2010.

Note also that MU-TERM 5.0 has a web interface that allows inexperienced users to prove automatically termination by means of the ‘automatic’ option. This is very convenient for teaching purposes, for instance. And, apart from MTT, it is the only termination tool that accepts programs in OBJ/*Maude* syntax.

Therefore, MU-TERM 5.0 is no more a tool for proving termination of CSR only: We can say now that it has evolved to become a powerful termination tool which is able to prove termination of a wide range of interesting properties

<sup>5</sup> See <http://www.lri.fr/~marche/termination-competition/2007/>, where only AProVE and MU-TERM participated, and <http://termcomp.uibk.ac.at/termcomp/> where there were three more tools in the competition: AProVE, Jambox [13] (only in the 2009 edition), and VMTL [27]. AProVE and MU-TERM solved the same number of examples but MU-TERM was much faster. The 2008 edition had only one participant: AProVE.

of rewriting with important applications to prove termination of programs in sophisticated rewriting-based programming languages like Maude or OBJ\*.

## References

1. Alarcón, B.: Innermost Termination of Context-Sensitive Rewriting. Master's thesis, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, Valencia, Spain (2008)
2. Alarcón, B., Gutiérrez, R., Iborra, J., Lucas, S.: Proving Termination of Context-Sensitive Rewriting with MU-TERM. *Electronic Notes in Theoretical Computer Science* 188, 105–115 (2007)
3. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. *Information and Computation* 208, 922–968 (2010)
4. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In: Wolter, F. (ed.) *Proc. of the 6th International Symposium on Frontiers of Combining Systems, FroCoS'07*. LNAI, vol. 4720, pp. 73–87. Springer-Verlag (2007)
5. Alarcón, B., Lucas, S., Meseguer, J.: A Dependency Pair Framework for *AVC*-Termination. In: Ölveczky, P. (ed.) *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*. LNCS, Springer-Verlag, to appear (2010)
6. Alarcón, B., Lucas, S., Navarro-Marset, R.: Proving Termination with Matrix Interpretations over the Reals. In: Geser, A., Waldmann, J. (eds.) *Proc. of the 10th International Workshop on Termination, WST'09*. pp. 12–15 (2009)
7. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. *Theoretical Computer Science* 236(1–2), 133–178 (2000)
8. Borralleras, C., Lucas, S., Navarro-Marset, R., Rodríguez-Carbonell, E., Rubio, A.: Solving Non-linear Polynomial Arithmetic via SAT Modulo Linear Arithmetic. In: Schmidt, R.A. (ed.) *Proc. of the 22th Conference on Automated Deduction, CADE'09*. LNAI, vol. 5663, pp. 294–305. Springer-Verlag (2009)
9. Borralleras, C., Lucas, S., Rubio, A.: Recursive Path Orderings can be Context-Sensitive. In: Voronkov, A. (ed.) *Proc. of the 18th Conference on Automated Deduction, CADE'02*. LNAI, vol. 2392, pp. 314–331. Springer-Verlag (2002)
10. Clavel, M., Durán, F., Eker, S., Lincoln, P., Martí-Oliet, N., Meseguer, J., Talcott, C.: *All About Maude – A High-Performance Logical Framework*, LNCS, vol. 4350. Springer-Verlag (2007)
11. Contejean, E., Marché, C., Monate, B., Urbain, X.: Proving Termination of Rewriting with CiME. In: Rubio, A. (ed.) *Proc. of the 6th International Workshop on Termination, WST'03*. pp. 71–73 (2003)
12. Durán, F., Lucas, S., Meseguer, J.: MTT: The Maude Termination Tool (System Description). In: Armando, A., Baumgartner, P., Dowek, G. (eds.) *Proc. of the 4th International Joint Conference on Automated Reasoning, IJCAR'08*. LNCS, vol. 5195, pp. 313–319. Springer-Verlag (2008)
13. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewriting, available at <http://joerg.endrullis.de/jambox.html> (2009)
14. Giesl, J., Middeldorp, A.: Innermost Termination of Context-Sensitive Rewriting. In: Ito, M., Toyama, M. (eds.) *Proc. of the 6th International Conference on Developments in Language Theory, DLT'02*. LNAI, vol. 2450, pp. 231–244. Springer-Verlag (2003)

15. Giesl, J., Middeldorp, A.: Transformation Techniques for Context-Sensitive Rewrite Systems. *Journal of Functional Programming* 14(4), 379–427 (2004)
16. Giesl, J., Schneider-Kamp, P., Thiemann, R.: AProVE 1.2: Automatic Termination Proofs in the Dependency Pair Framework. In: Furbach, U., Shankar, N. (eds.) *Proc. of the 3rd International Joint Conference on Automated Reasoning, IJCAR'06*. LNAI, vol. 4130, pp. 281–286. Springer-Verlag, available at <http://www-i2.informatik.rwth-aachen.de/AProVE> (2006)
17. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning* 37(3), 155–203 (2006)
18. Gnaedig, I.: Termination of Order-sorted Rewriting. In: Kirchner, H., Levi, G. (eds.) *Proc. of the 3rd International Conference on Algebraic and Logic Programming, ALP'92*. LNAI, vol. 632, pp. 37–52. Springer-Verlag (1992)
19. Gramlich, B., Lucas, S.: Modular Termination of Context-Sensitive Rewriting. In: *Proc. of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP'02*. pp. 50–61. ACM Press (2002)
20. Gutiérrez, R., Lucas, S.: Proving Termination in the Context-Sensitive Dependency Pair Framework. In: Ölveczky, P. (ed.) *Proc. of the 8th International Workshop on Rewriting Logic and its Applications, WRLA'10*. LNCS, Springer-Verlag, to appear (2010)
21. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming* 1998(1), 1–61 (1998)
22. Lucas, S.: Termination of Rewriting With Strategy Annotations. In: Nieuwenhuis, R., Voronkov, A. (eds.) *Proc. of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*. LNAI, vol. 2250, pp. 666–680. Springer-Verlag (2001)
23. Lucas, S.: MU-TERM: A Tool for Proving Termination of Context-Sensitive Rewriting. In: van Oostrom, V. (ed.) *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04*. LNCS, vol. 3091, pp. 200–209. Springer-Verlag, available at <http://zenon.dsic.upv.es/muterm/> (2004)
24. Lucas, S.: Polynomials over the Reals in Proofs of Termination: from Theory to Practice. *RAIRO Theoretical Informatics and Applications* 39(3), 547–586 (2005)
25. Lucas, S.: Proving Termination of Context-Sensitive Rewriting by Transformation. *Information and Computation* 204(12), 1782–1846 (2006)
26. Lucas, S., Meseguer, J.: Order-Sorted Dependency Pairs. In: Antoy, S., Albert, E. (eds.) *Proc. of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08*. pp. 108–119. ACM Press (2008)
27. Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In: Treinen, R. (ed.) *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*. LNCS, vol. 5595, pp. 285–294. Springer-Verlag (2009)
28. Thiemann, R., Sternagel, C.: Loops under Strategies. In: Treinen, R. (ed.) *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*. LNCS, vol. 5595, pp. 17–31. Springer-Verlag (2009)
29. Winkler, S., Sato, H., Middeldorp, A., Kurihara, M.: Optimizing  $\text{mkb}_{TT}$ . In: Lynch, C. (ed.) *Proc. of the 21st International Conference on Rewriting Techniques and Applications, RTA'10*. *Leibniz International Proceedings in Informatics (LIPIcs)*, vol. 6, pp. 373–384. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, <http://drops.dagstuhl.de/opus/volltexte/2010/2664> (2010)

---

## Extended Versions

### 8.14 Proving Termination in the Context-Sensitive Dependency Pair Framework

9. R. Gutiérrez and S. Lucas. **Proving Termination in the Context-Sensitive Dependency Pairs Framework**. Technical Report, DSIC, UPV, 2010.

## Proving Termination in the Context-Sensitive Dependency Pair Framework\*

Raúl Gutiérrez<sup>1</sup> and Salvador Lucas<sup>1</sup>

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, [19]), a *replacement map*  $\mu$  satisfying  $\mu(f) \subseteq \{1, \dots, \text{ar}(f)\}$  for every function symbol  $f$  of arity  $\text{ar}(f)$  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* [7], see [8, 10, 11].

---

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

2 Raúl Gutiérrez and Salvador Lucas

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

$$\begin{array}{ll}
 \text{gt}(0, y) \rightarrow \text{false} & \text{p}(0) \rightarrow 0 \\
 \text{gt}(s(x), 0) \rightarrow \text{true} & \text{p}(s(x)) \rightarrow x \\
 \text{gt}(s(x), s(y)) \rightarrow \text{gt}(x, y) & \text{minus}(x, y) \rightarrow \text{if}(\text{gt}(y, 0), \text{minus}(\text{p}(x), \text{p}(y)), x) \\
 \text{if}(\text{true}, x, y) \rightarrow x & \text{div}(0, s(y)) \rightarrow 0 \\
 \text{if}(\text{false}, x, y) \rightarrow y & \text{div}(s(x), s(y)) \rightarrow s(\text{div}(\text{minus}(x, y), s(y)))
 \end{array}$$

with  $\mu(\text{if}) = \{1\}$  and  $\mu(f) = \{1, \dots, \text{ar}(f)\}$  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:

$$\text{minus}(0, 0) \rightarrow_{\mathcal{R}}^* \text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0) \rightarrow_{\mathcal{R}}^* \dots \text{minus}(0, 0), \dots \rightarrow_{\mathcal{R}}^* \dots$$

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

In [2], Arts and Giesl's dependency pair approach [6], a powerful technique for proving termination of rewriting, was adapted to CSR (see [3] 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  $\ell \rightarrow r$  such that  $r$  contains some *defined* symbol<sup>1</sup>  $\mathbf{g}$ . Intuitively, we can think of these rules as representing possible (direct or indirect) recursive calls. Recursion paths associated to each rule  $\ell \rightarrow r$  are represented as new rules  $u \rightarrow v$  (called *dependency pairs*) where  $u = \mathbf{f}^\sharp(\ell_1, \dots, \ell_k)$  if  $\ell = f(\ell_1, \dots, \ell_k)$  and  $v = \mathbf{g}^\sharp(s_1, \dots, s_m)$  if  $s = \mathbf{g}(s_1, \dots, s_m)$  is a subterm of  $r$  and  $\mathbf{g}$  is a defined symbol. The notation  $\mathbf{f}^\sharp$  for a given symbol  $\mathbf{f}$  means that  $\mathbf{f}$  is *marked*. In practice, we often capitalize  $\mathbf{f}$  and use  $\mathbf{F}$  instead of  $\mathbf{f}^\sharp$  in our examples. For this reason, the dependency pair technique starts by considering a new TRS  $\text{DP}(\mathcal{R})$  which contains all these dependency pairs for each  $\ell \rightarrow r \in \mathcal{R}$ . The rules in  $\mathcal{R}$  and  $\text{DP}(\mathcal{R})$  determine the so-called *dependency chains* whose finiteness characterizes termination of  $\mathcal{R}$  [6]. 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  $\ell \rightarrow r$  which are considered to build the *context-sensitive dependency pairs*  $\ell^\sharp \rightarrow s^\sharp$  must be  $\mu$ -replacing terms. In sharp contrast with the dependency pair approach, though, we also need *collapsing dependency pairs*  $u \rightarrow x$  where  $u$  is obtained from the left-hand side  $\ell$  of a rule  $\ell \rightarrow r$  in the usual way, i.e.,  $u = \ell^\sharp$  but  $x$  is a *migrating variable* which is  $\mu$ -replacing in  $r$  but which only occurs at *non- $\mu$ -replacing positions* in  $\ell$  [2, 3]. 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

<sup>1</sup> A symbol  $\mathbf{g} \in \mathcal{F}$  is defined in  $\mathcal{R}$  if there is a rule  $\ell \rightarrow r$  in  $\mathcal{R}$  whose left-hand side  $\ell$  is of the form  $\mathbf{g}(\ell_1, \dots, \ell_k)$  for some  $k \geq 0$ .

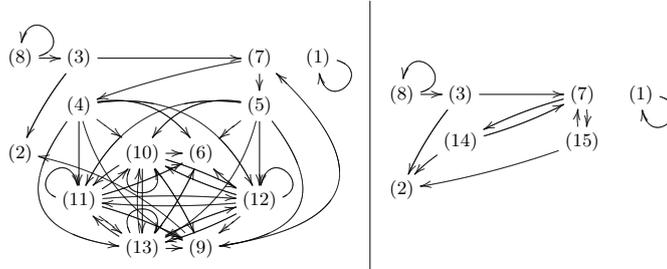


Fig. 1. Dependency graph for Example 1 following [1] (left) and [3] (right)

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

In [1], 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 lose 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 [1], we have the following dependency pairs (a new symbol U is introduced):*

$GT(s(x), s(y)) \rightarrow GT(x, y)$ (1)	$M(x, y) \rightarrow IF(gt(y, 0), minus(p(x), p(y)), x)$ (7)
$M(x, y) \rightarrow GT(y, 0)$ (2)	$D(s(x), s(y)) \rightarrow D(minus(x, y), s(y))$ (8)
$D(s(x), s(y)) \rightarrow M(x, y)$ (3)	$U(minus(p(x), p(y))) \rightarrow M(p(x), p(y))$ (9)
$IF(true, x, y) \rightarrow U(x)$ (4)	$U(p(x)) \rightarrow U(x)$ (10)
$IF(false, x, y) \rightarrow U(y)$ (5)	$U(p(y)) \rightarrow U(y)$ (11)
$U(p(x)) \rightarrow P(x)$ (6)	$U(minus(x, y)) \rightarrow U(x)$ (12)
	$U(minus(x, y)) \rightarrow U(y)$ (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 [2, 3], our set of dependency pairs is the following:

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

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

The work in [1] was motivated by the fact that mechanizing proofs of termination of CSR according to the results in [2] can be difficult due to the presence of collapsing dependency pairs. The problem is that [2] 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

4 Raúl Gutiérrez and Salvador Lucas

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  $\mu$ -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 [1]. Section 9 concludes. Proofs can be found in [15].

## 2 Preliminaries

We assume a basic knowledge about standard definitions and notations for term rewriting as given in, e.g., [22]. Positions  $p, q, \dots$  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  $\Lambda$ . The set of positions of a term  $t$  is  $\mathcal{P}os(t)$ . Positions of nonvariable symbols  $f \in \mathcal{F}$  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  $t \geq s$  if  $s = t|_p$  for some  $p \in \mathcal{P}os(t)$  and  $t \triangleright s$  if  $t \geq s$  and  $t \neq s$ . The symbol labeling the root of  $t$  is denoted as  $\mathit{root}(t)$ . A *substitution* is a mapping  $\sigma : \mathcal{X} \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$  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  $C \in \mathcal{T}(\mathcal{F} \cup \{\square\}, \mathcal{X})$  with a ‘hole’  $\square$  (a fresh constant symbol). A *rewrite rule* is an ordered pair  $(\ell, r)$ , written  $\ell \rightarrow r$ , with  $\ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ ,  $\ell \notin \mathcal{X}$  and  $\mathit{Var}(r) \subseteq \mathit{Var}(\ell)$ . The left-hand side (*lhs*) of the rule is  $\ell$  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  $c \in \mathcal{C}$ , called *constructors* and symbols  $f \in \mathcal{D}$ , called *defined symbols*, where  $\mathcal{D} = \{\mathit{root}(\ell) \mid \ell \rightarrow r \in R\}$  and  $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ .

In the following, we introduce some notions and notation about CSR [19]. A mapping  $\mu : \mathcal{F} \rightarrow \wp(\mathbb{N})$  is a *replacement map* if  $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \dots, \mathit{ar}(f)\}$ . 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  $\mathcal{P}os^{\mu}(t) = \{\Lambda\} \cup \bigcup_{i \in \mu(\mathit{root}(t))} i.\mathcal{P}os^{\mu}(t|_i)$ , if  $t \notin \mathcal{X}$ . The set of  $\mu$ -replacing variables of  $t$  is  $\mathit{Var}^{\mu}(t) = \{x \in \mathit{Var}(t) \mid \exists p \in \mathcal{P}os^{\mu}(t), t|_p = x\}$  and  $\mathit{Var}^{\#}(t) = \{x \in \mathit{Var}(t) \mid \exists p \in \mathcal{P}os(t) \setminus \mathcal{P}os^{\mu}(t), t|_p = x\}$  is the set of *non- $\mu$ -replacing variables* of  $t$ . Note that  $\mathit{Var}^{\mu}(t)$  and  $\mathit{Var}^{\#}(t)$  do not need to be disjoint. The  $\mu$ -replacing subterm relation  $\triangleright_{\mu}$  is given by  $t \triangleright_{\mu} s$  if there is  $p \in \mathcal{P}os^{\mu}(t)$  such that  $s = t|_p$ . We write  $t \triangleright_{\mu} s$  if  $t \triangleright_{\mu} s$  and  $t \neq s$ . We write

$t \triangleright_{\mu} s$  to denote that  $s$  is a *non- $\mu$ -replacing strict subterm* of  $t$ , i.e., there is a *non- $\mu$ -replacing position*  $p \in \mathcal{P}os(t) \setminus \mathcal{P}os^{\mu}(t)$  such that  $s = t|_p$ . In CSR, we (only) contract  *$\mu$ -replacing redexes*:  $t$   $\mu$ -rewrites to  $s$ , written  $t \hookrightarrow_{\mathcal{R},\mu} s$  (or  $t \xrightarrow{p}_{\mathcal{R},\mu} s$  to make position  $p$  explicit), iff there are  $\ell \rightarrow r \in R$ ,  $p \in \mathcal{P}os^{\mu}(t)$  and a substitution  $\sigma$  such that  $t|_p = \sigma(\ell)$  and  $s = t[\sigma(r)]_p$ ;  $t \xrightarrow{q}_{\mathcal{R},\mu} s$  means that the  $\mu$ -rewrite step is applied below position  $q$ , i.e.,  $p > q$ . We say that a variable  $x$  is *migrating* in  $\ell \rightarrow r \in R$  if  $x \in \mathcal{V}ar^{\mu}(r) \setminus \mathcal{V}ar^{\mu}(\ell)$ . A term  $t$  is  *$\mu$ -terminating* if there is no infinite  $\mu$ -rewrite sequence  $t = t_1 \hookrightarrow_{\mathcal{R},\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} \dots \hookrightarrow_{\mathcal{R},\mu} t_n \hookrightarrow_{\mathcal{R},\mu} \dots$  starting from  $t$ . A TRS  $\mathcal{R} = (\mathcal{F}, R)$  is  *$\mu$ -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 $\mu$ -Rewrite Sequences

Let  $\mathcal{M}_{\infty,\mu}$  be a set of *minimal non- $\mu$ -terminating terms* in the following sense:  $t$  belongs to  $\mathcal{M}_{\infty,\mu}$  if  $t$  is non- $\mu$ -terminating and every strict  *$\mu$ -replacing subterm*  $s$  of  $t$  (i.e.,  $t \triangleright_{\mu} s$ ) is  $\mu$ -terminating [2]. Minimal terms allow us to characterize infinite  $\mu$ -rewrite sequences [3]. In [3], 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 [3]).** *Let  $\mathcal{T}_{\infty,\mu}$  be a set of strongly minimal non- $\mu$ -terminating terms in the following sense:  $t$  belongs to  $\mathcal{T}_{\infty,\mu}$  if  $t$  is non- $\mu$ -terminating and every strict subterm  $u$  (i.e.,  $t \triangleright u$ ) is  $\mu$ -terminating. It is obvious that  $\mathit{root}(t) \in \mathcal{D}$  for all  $t \in \mathcal{T}_{\infty,\mu}$ .*

Every non- $\mu$ -terminating term has a subterm that is strongly minimal. Then, given a non- $\mu$ -terminating term  $t$  we can always find a subterm  $t_0 \in \mathcal{T}_{\infty,\mu}$  of  $t$  which starts a *minimal infinite  $\mu$ -rewrite sequence* of the form  $t_0 \xrightarrow{\geq}_{\mathcal{R},\mu}^* \sigma_1(\ell_1) \xrightarrow{\Delta}_{\mathcal{R},\mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq}_{\mathcal{R},\mu}^* \sigma_2(\ell_2) \xrightarrow{\Delta}_{\mathcal{R},\mu} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{\geq}_{\mathcal{R},\mu}^* \dots$  where  $t_i, \sigma_i(\ell_i) \in \mathcal{M}_{\infty,\mu}$  for all  $i > 0$  [3]. Theorem 1 below tells us that we have two possibilities:

- The minimal non- $\mu$ -terminating terms  $t_i \in \mathcal{M}_{\infty,\mu}$  in the sequence are partially introduced by a  $\mu$ -replacing nonvariable subterm of the right-hand sides  $r_i$  of the rules  $\ell_i \rightarrow r_i$ .
- The minimal non- $\mu$ -terminating terms  $t_i \in \mathcal{M}_{\infty,\mu}$  in the sequence are introduced by instantiated *migrating variables*  $x_i$  of (the respective) rules  $\ell_i \rightarrow r_i$ , 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- $\mu$ -replacing positions in the right-hand sides of the rules (*hidden terms*) within a given (*hiding*) context.

We use the following functions [2, 3]:  $\mathit{REN}^{\mu}(t)$ , which *independently* renames all *occurrences* of  $\mu$ -replacing variables by using new fresh variables which are not

6 Raúl Gutiérrez and Salvador Lucas

in  $\mathcal{V}ar(t)$ , and  $\text{NARR}_{\mathcal{R}}^{\mu}(t)$ , which indicates whether  $t$  is  $\mu$ -narrowable<sup>2</sup> (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* [1, 3] if there is a rule  $\ell \rightarrow r \in R$  such that  $t$  is a non- $\mu$ -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  $\mu$ -narrowable hidden terms headed by a defined symbol:

$$\mathcal{NHT}(\mathcal{R}, \mu) = \{t \in \mathcal{HT}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D} \text{ and } \text{NARR}_{\mathcal{R}}^{\mu}(\text{REN}^{\mu}(t))\}$$

**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  $\ell \rightarrow r \in \mathcal{R}$  if  $r \triangleright_{\mu} f(r_1, \dots, r_n)$  for some terms  $r_1, \dots, r_n$ , and there is  $i \in \mu(f)$  such that  $r_i$  contains a  $\mu$ -replacing defined symbol (i.e.,  $\text{Pos}_{\mathcal{D}}^{\mu}(r_i) \neq \emptyset$ ) or a variable  $x \in (\mathcal{V}ar^{\#}(\ell) \cap \mathcal{V}ar^{\#}(r)) \setminus (\mathcal{V}ar^{\mu}(\ell) \cup \mathcal{V}ar^{\mu}(r))$  which is  $\mu$ -replacing in  $r_i$  (i.e.,  $x \in \mathcal{V}ar^{\mu}(r_i)$ ). A context  $C[\square]$  is *hiding [1]* if  $C[\square] = \square$ , or  $C[\square] = f(t_1, \dots, t_{i-1}, C'[\square], t_{i+1}, \dots, t_k)$ , where  $f$  hides position  $i$  and  $C'[\square]$  is a hiding context.*

Definition 2 is a refinement of [1, Definition 7], where the new condition  $x \in (\mathcal{V}ar^{\#}(\ell) \cap \mathcal{V}ar^{\#}(r)) \setminus (\mathcal{V}ar^{\mu}(\ell) \cup \mathcal{V}ar^{\mu}(r))$  is useful to discard contexts that are not valid when minimality is considered.

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

These notions are used and combined to model infinite context-sensitive rewrite sequences starting from strongly minimal non- $\mu$ -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*

$$t = t_0 \xrightarrow{\geq_{\mathcal{R}, \mu}^A} \sigma_1(\ell_1) \xrightarrow{A} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq_{\mathcal{R}, \mu}^A} \sigma_2(\ell_2) \xrightarrow{A} \dots$$

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

1.  $t_i = \sigma_i(s_i)$  for some nonvariable term  $s_i$  such that  $r_i \triangleright_{\mu} s_i$ , 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  $C_i[\square]$ , and substitution  $\theta_i$ .

## 4 Chains of Context-Sensitive Dependency Pairs

In this section, we revise the definition of chain of context-sensitive dependency pairs given in [3]. First, we recall the notion of context-sensitive dependency pair.

<sup>2</sup> A term  $s$   $\mu$ -narrows to the term  $t$  if there is a nonvariable position  $p \in \text{Pos}_{\mathcal{F}}^{\mu}(s)$  and a rule  $\ell \rightarrow r$  such that  $s|_p$  and  $\ell$  unify with mgu  $\sigma$ , and  $t = \sigma(s[r]_p)$ .

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

$$\begin{aligned} \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \triangleright_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\triangleright_\mu s, \text{NARR}_{\mathcal{R}}^\mu(\text{REN}^\mu(s))\} \\ \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) &= \{\ell^\sharp \rightarrow x \mid \ell \rightarrow r \in R, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell)\} \end{aligned}$$

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

Now, we provide a new notion of *chain* of CSDPs. In contrast to [1], we store the information about hidden terms and hiding contexts which is relevant to model infinite minimal  $\mu$ -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  $\text{unh}(\mathcal{R}, \mu)$  as the TRS consisting of the following rules:

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

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

$$\begin{array}{ll} \text{minus}(p(x), p(y)) \rightarrow M(p(x), p(y)) & (16) \quad \text{minus}(x, y) \rightarrow y & (18) \\ p(x) \rightarrow P(x) & (17) \quad \text{minus}(x, y) \rightarrow x & (19) \end{array}$$

Definitions 3 and 4 lead to a suitable notion of *chain* which captures minimal infinite  $\mu$ -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  $s \rightarrow t \in \mathcal{S}$  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  $u_i \rightarrow v_i \in \mathcal{P}$ , together with a substitution  $\sigma$  satisfying that, for all  $i \geq 1$ ,

1. if  $v_i \notin \text{Var}(u_i) \setminus \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) = t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ , and
2. if  $v_i \in \text{Var}(u_i) \setminus \text{Var}^\mu(u_i)$ , then  $\sigma(v_i) \xrightarrow{\mathcal{S}_{\triangleright_\mu, \mu}}^* \circ \xrightarrow{\mathcal{A}}_{\mathcal{S}_{\triangleright_\mu, \mu}} t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ .

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  $f(x_1, \dots, x_k) \rightarrow x_i$  for all  $f \in \mathcal{D}$  and  $i \in \mu(f)$  (where  $x_1, \dots, x_k$  are variables) are used in Item 1 of Definition 4, then Definition 5 yields the notion of chain in [3]; and if, additionally, rules  $f(x_1, \dots, x_k) \rightarrow f^\sharp(x_1, \dots, x_k)$  for all  $f \in \mathcal{D}$  are used in Item 2 of Definition 4, then we have the original notion of chain in [2]. 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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ -chain.

8 Raúl Gutiérrez and Salvador Lucas

## 5 Context-Sensitive Dependency Pair Framework

In the DP framework [13], 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 [1, 3]).

**Definition 6 (CS Problem and CS Processor).** A CS problem  $\tau$  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- $\mu$ -terminating or there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain.

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

In order to prove the  $\mu$ -termination of a TRS  $\mathcal{R}$ , we adapt the result from [13] 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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ . For every inner node labeled with  $\tau$ , there is a sound processor  $\text{Proc}$  satisfying one of the following conditions:

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

If all leaves of the tree are labeled with “yes”, then  $\mathcal{R}$  is  $\mu$ -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- $\mu$ -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 [1] into our framework. The pairs in a CS-TRS  $(\mathcal{P}, \mu)$ , where  $\mathcal{P} = (\mathcal{G}, P)$ , are partitioned as follows:  $P_{\mathcal{X}} = \{u \rightarrow v \in P \mid v \in \mathcal{V}ar(u) \setminus \mathcal{V}ar^\mu(u)\}$  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  $P_{\cup}$  be given by the following rules:

- $u \rightarrow U(x)$  for every  $u \rightarrow x \in P_{\mathcal{X}}$ ,
- $U(s) \rightarrow U(t)$  for every  $s \rightarrow t \in \mathcal{S}_{>\mu}$ , and

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

Here,  $U$  is a new fresh symbol. Let  $\mathcal{P}' = (\mathcal{G} \cup \{U\}, P')$  where  $P' = (P \setminus P_{\mathcal{X}}) \cup P_U$ , and  $\mu'$  extends  $\mu$  by  $\mu'(U) = \emptyset$ . The processor  $\text{Proc}_{e\text{Coll}}$  given by  $\text{Proc}_{e\text{Coll}}(\tau) = \{(\mathcal{P}', \mathcal{R}, \emptyset, \mu')\}$  is sound and complete.

Now, we can apply all CS processors from [1] and [3] which did not consider any  $S$  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 [3]). For instance, we can use the processor in Theorem 5 below, which is often applied in proofs of termination of CSR with MU-TERM [4]. The subTRS of  $\mathcal{P}_{\mathcal{X}}$  containing the rules whose migrating variables occur on non- $\mu$ -replacing immediate subterms in the left-hand side is  $\mathcal{P}_{\mathcal{X}}^1 = \{f(u_1, \dots, u_k) \rightarrow x \in \mathcal{P}_{\mathcal{X}} \mid \exists i, 1 \leq i \leq k, i \notin \mu(f), x \in \text{Var}(u_i)\}$ .

**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  $s \rightarrow t \in \mathcal{S}_\sharp$ ,  $t \notin \mathcal{X}$  (2)  $\{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_\sharp\} \cap \mathcal{D} = \emptyset$  and (3) for all  $s \rightarrow t \in \mathcal{S}_\sharp$ , we have that  $s = f(s_1, \dots, s_k)$  and  $t = g(s_1, \dots, s_k)$  for some  $k \in \mathbb{N}$ , function symbols  $f, g \in \mathcal{H}$ , and terms  $s_1, \dots, s_k$ . Then, the processors  $\text{Proc}_{\text{CollI}}$  given by*

$$\text{Proc}_{\text{CollI}}(\tau) = \begin{cases} \emptyset & \text{if } \mathcal{P} = \mathcal{P}_{\mathcal{X}}^1 \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

*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  $\text{Proc}_{\text{CollI}}(\tau)$  to conclude that the CS problem  $\tau$  is finite.*

## 5.2 Context-Sensitive Dependency Graph

In the DP-approach [6, 13], a *dependency graph* is associated to the TRS  $\mathcal{R}$ . The nodes of the graph are the dependency pairs in  $\text{DP}(\mathcal{R})$  and there is an arc from a dependency pair  $u \rightarrow v$  to a dependency pair  $u' \rightarrow v'$  if there are substitutions  $\theta$  and  $\theta'$  such that  $\theta(v) \rightarrow_{\mathcal{R}}^* \theta'(u')$ . 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  $G(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  has  $\mathcal{P}$  as the set of nodes. Given  $u \rightarrow v, u' \rightarrow v' \in \mathcal{P}$ , there is an arc from  $u \rightarrow v$  to  $u' \rightarrow v'$  if  $u \rightarrow v, u' \rightarrow v'$  is a minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain for some substitution  $\sigma$ .*

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) [18]. The following result formalizes the use of SCCs for dealing with CS problems.

10 Raúl Gutiérrez and Salvador Lucas

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

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

(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  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$ , which renames all subterms headed by a ‘defined’ symbol in  $\mathcal{R}$  by new fresh variables if it can be rewritten:

**Definition 8 ( $\text{TCAP}_{\mathcal{R}}^{\mu}$  [3]).** *Given a TRS  $\mathcal{R}$  and a replacement map  $\mu$ , we let  $\text{TCAP}_{\mathcal{R}}^{\mu}$  be as follows:*

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

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

where  $y$  is a new fresh variable,  $[s]_i^f = \text{TCAP}_{\mathcal{R}}^{\mu}(s)$  if  $i \in \mu(f)$  and  $[s]_i^f = s$  if  $i \notin \mu(f)$ . We assume that  $\ell$  shares no variable with  $f([t_1]_1^f, \dots, [t_k]_k^f)$  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  $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ ) has  $\mathcal{P}$  as the set of nodes and arcs which connect them as follows:*

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

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

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

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

where

- $\mathcal{S}_{\mathcal{Q}} = \emptyset$  if  $\mathcal{Q}_{\mathcal{X}} = \emptyset$ .

- $\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow t \mid s \rightarrow t \in \mathcal{S}_{\sharp}, \text{TCAP}_{\mathcal{R}}^{\mu}(t) \text{ and } u' \text{ unify for some } u' \rightarrow v' \in \mathcal{Q}\}$  if  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ .

is sound and complete.

*Example 6.* In Figure 1 (right) we show  $\text{EG}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$  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  $\text{Proc}_{\text{SCC}}$  to the initial CS problem  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$  for  $(\mathcal{R}, \mu)$  in Example 1, then we obtain the problems:  $(\mathcal{P}_1, \mathcal{R}, \emptyset, \mu^{\sharp})$ ,  $(\mathcal{P}_2, \mathcal{R}, \emptyset, \mu^{\sharp})$ ,  $(\mathcal{P}_3, \mathcal{R}, \mathcal{S}_3, \mu^{\sharp})$ , where  $\mathcal{S}_3 = \{(16), (18), (19)\}$ .

### 5.3 Reduction Triple Processor

A  $\mu$ -reduction pair  $(\succsim, \sqsupset)$  consists of a stable and  $\mu$ -monotonic<sup>3</sup> quasi-ordering  $\succsim$ , and a well-founded stable relation  $\sqsupset$  on terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  which are compatible, i.e.,  $\succsim \circ \sqsupset \subseteq \sqsupset$  or  $\sqsupset \circ \succsim \subseteq \sqsupset$  [2].

In [2, 3], when a collapsing pair  $u \rightarrow x$  occurs in a chain, we have to *look inside* the instantiated right-hand side  $\sigma(x)$  for a  $\mu$ -replacing subterm that, after marking it, does  $\mu$ -rewrite to the (instantiated) left-hand side of another pair. For this reason, the quasi-orderings  $\succsim$  of reduction pairs  $(\succsim, \sqsupset)$  which are used in [2, 3] are required to have the  $\mu$ -subterm property, i.e.  $\sqsupset_{\mu} \subseteq \succsim$ . This is equivalent to impose  $f(x_1, \dots, x_k) \succsim x_i$  for all projection rules  $f(x_1, \dots, x_k) \rightarrow x_i$  with  $f \in \mathcal{F}$  and  $i \in \mu(f)$ . This is similar for markings: in [2] we have to ensure that  $f(x_1, \dots, x_k) \succsim^{\sharp} f^{\sharp}(x_1, \dots, x_k)$  for all defined symbols  $f$  in the signature. In [3], thanks to the notion of hidden term, we relaxed the last condition: we require  $t \succsim t^{\sharp}$  for all (narrowable) *hidden terms*  $t$ . In [1], thanks to the notion of hiding context, we only require that  $\succsim$  is compatible with the projections  $f(x_1, \dots, x_k) \rightarrow x_i$  for those symbols  $f$  and positions  $i$  such that  $f$  *hides position*  $i$ . However, this information is implicitly encoded as (new) pairs  $\text{U}(f(x_1, \dots, x_k)) \rightarrow \text{U}(x_i)$  in the set  $\mathcal{P}$ . The strict component  $\sqsupset$  of the reduction pair  $(\succsim, \sqsupset)$  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 [1, 3]) 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  $\mu$ -monotonicity is required).

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

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

<sup>3</sup> A binary relation  $R$  on terms is  $\mu$ -monotonic if for all terms  $s, t, t_1, \dots, t_k$ , and  $k$ -ary symbols  $f$ , whenever  $s R t$  and  $i \in \mu(f)$  we have  $f(t_1, \dots, t_{i-1}, s, \dots, t_k) R f(t_1, \dots, t_{i-1}, t, \dots, t_k)$ .

12 Raúl Gutiérrez and Salvador Lucas

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

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

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

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  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ . Another advantage is that we can now *remove rules from  $\mathcal{S}$* .

#### 5.4 Reduction Triple Processor with Usable Rules

In order to use  $\text{Proc}_{RT}$ , we require that the rules in the TRS  $\mathcal{R}$  of the CS problem  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  are included in  $\succcurlyeq$ , i.e.,  $\mathcal{R} \subseteq \succcurlyeq$  must hold. However, it would be desirable to consider only the rules which are really necessary to capture all possible infinite sequences instead of *all* rules  $\mathcal{R}$  in the CS problem. *Usable rules* [6, 17, 24] provide a sound estimation of this ‘minimal’ set.

Usable rules were introduced by Arts and Giesl in [6] in connection with innermost termination. Hirokawa and Middeldorp [17] and (independently) Thiemann et al. [24] showed how to use them to prove termination of rewriting.

In order to adapt the notion of usable rules to CS problems, we adapt the approach followed in [16], that is based on the notion of *dependency* among function symbols. Let  $\text{Rls}_{\mathcal{R}}(f) = \{\ell \rightarrow r \in \mathcal{R} \mid \text{root}(\ell) = f\}$ . The set of  $\mu$ -replacing symbols in a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is denoted by  $\text{Fun}^{\mu}(t) = \{f \mid \exists p \in \text{Pos}^{\mu}(t), f = \text{root}(t|_p)\}$ . The simplest adaptation of this notion is the following.

**Definition 10 (Basic  $\mu$ -Dependency [16]).** *Given a TRS  $(\mathcal{F}, R)$  and  $\mu \in M_{\mathcal{F}}$ , we say that  $f \in \mathcal{F}$  has a basic  $\mu$ -dependency on  $h \in \mathcal{F}$  (written  $f \blacktriangleright_{\mathcal{R}, \mu} h$ ) if  $f = h$  or there is a function symbol  $g$  with  $g \blacktriangleright_{\mathcal{R}, \mu} h$  and a rule  $\ell \rightarrow r \in \text{Rls}_{\mathcal{R}}(f)$  with  $g \in \text{Fun}^{\mu}(r)$ .*

The corresponding notion of *basic CS usable rule* is the following.

**Definition 11 (Basic CS Usable Rules).** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. The set  $\mathcal{U}^{\blacktriangleright}(\tau)$  of basic context-sensitive usable rules of  $\tau$  is*

$$\mathcal{U}^{\blacktriangleright}(\tau) = \bigcup_{u \rightarrow v \in \mathcal{P}, f \in \text{Fun}^{\mu}(v), f \blacktriangleright_{\mathcal{R}, \mu} g} \text{Rls}_{\mathcal{R}}(g)$$

However, Definition 11 does *not* lead to a correct approach for proving termination of CSR.

*Example 7.* Consider the TRS  $\mathcal{R}$  [5]:

$$\begin{array}{l} f(c(x), x) \rightarrow f(x, x) \\ \mathbf{b} \rightarrow c(\mathbf{b}) \end{array}$$

together with  $\mu(f) = \{1, 2\}$  and  $\mu(c) = \emptyset$ . We have the following set of CSDPs  $\text{DP}(\mathcal{R}, \mu)$ :

$$F(c(x), x) \rightarrow F(x, x)$$

The un hiding TRS  $\text{unh}(\mathcal{R}, \mu)$  is:

$$\mathbf{b} \rightarrow \mathbf{B}$$

Since there is no collapsing pair in  $\text{DP}(\mathcal{R}, \mu)$ , after applying  $\text{Proc}_{\text{SCC}}$  to  $\tau_0 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ , we obtain

$$\tau_1 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\sharp)$$

According to Definition 11, we have no basic usable rules for  $\tau$  because  $F(x, x)$  contains no symbol in  $\mathcal{F}$ . We could wrongly conclude finiteness of  $\tau_1$  the CS problem and, hence,  $\mu$ -termination of  $(\mathcal{R}, \mu)$ , but we have the following infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\sharp)$ -chain where  $\mathbf{b} \rightarrow c(\mathbf{b})$  is *used*:

$$\underline{F(c(\mathbf{b}), \mathbf{b})} \xrightarrow{\mathcal{P}, \mu^\sharp} \underline{F(\mathbf{b}, \mathbf{b})} \xrightarrow{\mathcal{R}, \mu^\sharp} \underline{F(c(\mathbf{b}), \mathbf{b})} \xrightarrow{\mathcal{P}, \mu^\sharp} \dots$$

As we show below, although basic usable rules are not correct for all CS problems, they can be used in presence of *strong conservativity*.

**Definition 12 (Strong Conservativity [16]).** *Let  $\mathcal{R}$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . A rule  $\ell \rightarrow r$  is strongly  $\mu$ -conservative if it is  $\mu$ -conservative and  $\text{Var}^\mu(\ell) \cap \text{Var}^\mu(r) = \text{Var}^\mu(r) \cap \text{Var}^\mu(\ell) = \emptyset$ ; and  $\mathcal{R}$  is strongly  $\mu$ -conservative if all rules in  $\mathcal{R}$  are strongly  $\mu$ -conservative.*

In order to obtain an appropriate and general definition of usable rule for CSR we have to consider the rules of *symbols* in hidden terms (that is, the rules of *hidden symbols as usable*). We also extend the notion of  $\mu$ -dependency to capture the usable rules when collapsing pairs are present:

*Example 8.* Consider the following CS problem  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  where  $\mathcal{P}$  is:

$$\begin{array}{l} \mathbf{G}(\mathbf{a}) \rightarrow \mathbf{F}(\mathbf{g}(\mathbf{b})) \\ \mathbf{F}(x) \rightarrow x \end{array}$$

the only rule in  $\mathcal{R}$  is:

$$\mathbf{b} \rightarrow \mathbf{a}$$

and  $\mathcal{S}$  consists of a single rule as well:

$$\mathbf{g}(x) \rightarrow \mathbf{G}(x)$$

Let  $\mu$  be given by  $\mu(\mathbf{g}) = \mu(\mathbf{G}) = \{1\}$  and  $\mu(\mathbf{F}) = \mu(\mathbf{a}) = \mu(\mathbf{b}) = \emptyset$ . Since we have a collapsing pair, the rule in  $\mathcal{R}$  is usable because it is necessary to build the following infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain:

$$\underline{\mathbf{G}(\mathbf{a})} \xrightarrow{\mathcal{P}, \mu} \underline{\mathbf{F}(\mathbf{g}(\mathbf{b}))} \xrightarrow{\mathcal{P}, \mu} \circ \xrightarrow{\mathcal{S}, \mu} \underline{\mathbf{G}(\mathbf{b})} \xrightarrow{\mathcal{R}, \mu} \underline{\mathbf{G}(\mathbf{a})} \xrightarrow{\mathcal{P}, \mu} \dots$$

14 Raúl Gutiérrez and Salvador Lucas

But, even taking into account the hidden symbols and extending the notion of  $\mu$ -dependency, we do not get a correct definition of usable rule:

*Example 9.* Consider the following ( $\mu$ -conservative) non- $\mu$ -terminating CS-TRS  $\mathcal{R}$ :

$$\begin{aligned} a(x, y) &\rightarrow b(x, x) \\ d(x, e) &\rightarrow a(x, x) \\ b(x, g) &\rightarrow d(x, x) \\ g &\rightarrow e \end{aligned}$$

with  $\mu(a) = \mu(d) = \{1, 2\}$ ,  $\mu(b) = \{1\}$  and  $\mu(g) = \mu(e) = \emptyset$ . The set  $DP(\mathcal{R}, \mu)$  of CSDPs is:

$$\begin{aligned} A(x, y) &\rightarrow B(x, x) \\ D(x, e) &\rightarrow A(x, x) \\ B(x, g) &\rightarrow D(x, x) \end{aligned}$$

and, since  $\text{unh}(\mathcal{R}, \mu)$  is empty, finiteness of the following CS problem:

$$\tau = (DP(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^\sharp)$$

is equivalent to  $\mu$ -termination of  $\mathcal{R}$ . According to Definition 11, we have no basic usable rules because the right-hand sides of the dependency pairs have no defined symbols and we have no hidden symbol since there is no hidden term.

In order to use the usable rules instead of all the rules, we add to  $\mathcal{R}$  the following ( $\mathcal{C}_\varepsilon$ ) rules:

$$\begin{aligned} c(x, y) &\rightarrow x \\ c(x, y) &\rightarrow y \end{aligned}$$

which simulate the application of the “removed” rules. In this way, if we consider no replacement restrictions, the rule  $g \rightarrow e$  is not needed to capture the following infinite chain:

$$\underline{A(g, g)} \rightarrow_{\mathcal{P}} \underline{B(g, g)} \rightarrow_{\mathcal{P}} D(g, g) \rightarrow_{\mathcal{R}} \underline{D(g, e)} \rightarrow_{\mathcal{P}} \underline{A(g, g)} \rightarrow_{\mathcal{P}} \dots$$

because we have the following sequence using  $\mathcal{C}_\varepsilon$ -rules instead:

$$\begin{aligned} \underline{A(c(g, e), c(g, e))} &\rightarrow_{\mathcal{P}} \underline{B(c(g, e), c(g, e))} \rightarrow_{\mathcal{C}_\varepsilon} \underline{B(c(g, e), g)} \rightarrow_{\mathcal{P}} \\ \underline{D(c(g, e), c(g, e))} &\rightarrow_{\mathcal{C}_\varepsilon} \underline{D(c(g, e), e)} \rightarrow_{\mathcal{P}} \underline{A(c(g, e), c(g, e))} \rightarrow_{\mathcal{P}} \dots \end{aligned}$$

However, if we consider the replacement restrictions now, the  $\mu$ -rewrite step

$$\underline{B(c(g, e), c(g, e))} \hookrightarrow_{\mathcal{C}_\varepsilon} \underline{B(c(g, e), g)}$$

is not longer possible. Since the infinite sequence above can be regarded as an infinite  $\mu$ -rewrite sequence:

$$\underline{A(g, g)} \hookrightarrow_{\mathcal{P}, \mu} \underline{B(g, g)} \hookrightarrow_{\mathcal{P}, \mu} D(g, g) \hookrightarrow_{\mathcal{R}, \mu} \underline{D(g, e)} \hookrightarrow_{\mathcal{P}, \mu} \underline{A(g, g)} \hookrightarrow_{\mathcal{P}, \mu} \dots$$

In order to avoid this problem, we modify Definition 10 to take into account symbols occurring at frozen positions in the *left-hand sides* of the rules<sup>4</sup>. The set of *non- $\mu$ -replacing* symbols in a term  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$  is denoted by  $\text{Fun}^\mu(t) = \{\mathbf{f} \mid \exists p \in \text{Pos}(t) \setminus \text{Pos}^\mu(t), \mathbf{f} = \text{root}(t|_p)\}$ .

<sup>4</sup> A more detailed analysis can be found in [16]

**Definition 13 ( $\mu$ -Dependency [16]).** Given a TRS  $(\mathcal{F}, R)$  and  $\mu \in M_{\mathcal{F}}$ , we say that  $f \in \mathcal{F}$  has a  $\mu$ -dependency on  $h \in \mathcal{F}$ , written  $f \triangleright_{\mathcal{R}, \mu} h$ , if  $f = h$  or there is a function symbol  $g$  with  $g \triangleright_{\mathcal{R}, \mu} h$  and a rule  $\ell \rightarrow r \in Rls_{\mathcal{R}}(f)$  with  $g \in Fun^{\#}(\ell) \cup Fun(r)$ .

We adapt the notion of usable rules to obtain a new notion which can be used to deal with any kind of CS problems.

**Definition 14 (CS Usable Rules).** Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. The set  $\mathcal{U}^{\mathcal{P}}(\tau)$  of context-sensitive usable rules of  $\tau$  is

$$\begin{aligned} \mathcal{U}^{\mathcal{P}}(\tau) = & \bigcup_{u \rightarrow v \in \mathcal{P}, f \in Fun^{\#}(u) \cup Fun(v), f \triangleright_{\mathcal{R}, \mu} g} Rls_{\mathcal{R}}(g) \cup \\ & \bigcup_{\ell \rightarrow r \in \mathcal{R}, f \in Fun^{\#}(r), f \triangleright_{\mathcal{R}, \mu} g} Rls_{\mathcal{R}}(g) \cup \\ & \bigcup_{s \rightarrow t \in \mathcal{S}, f \in Fun(s) \cup Fun(t), f \triangleright_{\mathcal{R}, \mu} g} Rls_{\mathcal{R}}(g) \end{aligned}$$

Now, we can define a valid processor for  $\mu$ -reduction triples with usable rules. In order to formulate it, we have to consider the so-called  $\mathcal{C}_{\varepsilon}$ -compatibility of  $\succsim$ , i.e.,  $c(x, y) \succsim x$  and  $c(x, y) \succsim y$  holds for a fresh function symbol  $c$  [14].

**Theorem 10 ( $\mu$ -Reduction Triple Processor with Usable Rules).** Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem. Let  $(\succsim, \sqsupset, \succ)$  be a  $\mu$ -reduction triple such that

1.  $\mathcal{P} \subseteq \succsim \cup \sqsupset$ ,
2. at least one of the following holds:
  - (a)  $\mathcal{U}^{\mathcal{P}}(\tau) \subseteq \succsim$ ,  $\mathcal{P} \cup \mathcal{U}^{\mathcal{P}}(\tau)$  is strongly  $\mu$ -conservative,  $\succsim$  is  $\mathcal{C}_{\varepsilon}$ -compatible
  - (b)  $\mathcal{U}^{\mathcal{P}}(\tau) \subseteq \succsim$ ,  $\succsim$  is  $\mathcal{C}_{\varepsilon}$ -compatible,
  - (c)  $\mathcal{R} \subseteq \succsim$ ,
3. and, whenever  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$  we have that  $\mathcal{S} \subseteq \succsim \cup \sqsupset \cup \succ$ .

Let  $\mathcal{P}_{\sqsupset} = \{u \rightarrow v \in \mathcal{P} \mid u \sqsupset v\}$  and  $\mathcal{S}_{\sqsupset} = \{s \rightarrow t \in \mathcal{S} \mid s \sqsupset t\}$ . Then, the processor  $\text{Proc}_{UR}$  given by

$$\text{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\sqsupset}, \mu)\} & \text{if (1), (2) and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

*Example 10.* (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  $\text{Proc}_{UR}$  to the CS problem  $\tau$  by using the  $\mu$ -reduction triple  $(\geq, >, \geq)$  where  $\geq$  and  $>$  are the orderings induced by the following polynomial interpretation (see [20] for missing notation and definitions):

$$\begin{array}{ll} [\text{if}](x, y, z) = \frac{1}{2}x + y + z & [\text{minus}](x, y) = 2x + 2 + \frac{1}{2} \\ [\text{p}](x) = \frac{1}{2}x & [0] = 0 \\ [\text{false}] = 0 & [\text{s}](x) = 2x + 2 \\ [\text{true}] = 2 & [\text{gt}](x, y) = 2x + \frac{1}{2}y \\ [\text{M}](x, y) = 2x + 2y + \frac{1}{2} & [\text{IF}](x, y, z) = \frac{1}{2}x + y + z \end{array}$$

16 Raúl Gutiérrez and Salvador Lucas

then, we have  $[\ell] \geq [r]$  for all (usable) rules in  $\mathcal{R}$  (all the rules are usable except the (div)-rules because (7) is not strongly conservative) and, for the rules in  $\mathcal{P}_3$  and  $\mathcal{S}_3$ , we have

$$\begin{array}{l} [\mathbb{M}(x, y)] \geq [\mathbb{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x)] \quad [\text{minus}(p(x), p(y))] \geq [\mathbb{M}(p(x), p(y))] \\ [\mathbb{IF}(\text{true}, x, y)] > [x] \quad [\text{minus}(x, y)] > [y] \\ [\mathbb{IF}(\text{false}, x, y)] \geq [y] \quad [\text{minus}(x, y)] > [x] \end{array}$$

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

Furthermore, we can increase the power of this definition by using argument filterings [3].

### 5.5 Subterm Processor

The subterm criterion was adapted to CSR in [2], but its use was restricted to noncollapsing pairs [2, Theorem 5]. In [3], a new version for collapsing pairs was defined, but in this version you can only remove all collapsing pairs and the projection  $\pi$  is restricted to  $\mu$ -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 15 (Root Symbols of a TRS [3]).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS. The set of root symbols associated to  $\mathcal{R}$  is:

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

**Definition 16 (Simple Projection).** Let  $\mathcal{R}$  be a TRS. A simple projection for  $\mathcal{R}$  is a mapping  $\pi$  that assigns to every  $k$ -ary symbol  $f \in \text{Root}(\mathcal{R})$  an argument position  $i \in \{1, \dots, k\}$ . This mapping is extended to terms by

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

**Theorem 11 (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)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , then for all  $s \rightarrow t \in \mathcal{S}_{\sharp}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\sharp}\}$ . Let  $\mathcal{P}_{\pi, \triangleright_{\mu}} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright_{\mu} \pi(v)\}$  and  $\mathcal{S}_{\pi, \triangleright_{\mu}} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow t \in \mathcal{S}_{\sharp} \mid s \triangleright_{\mu} \pi(t)\}$ . Then,  $\text{Proc}_{\text{subterm}}$  given by

$$\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright_{\mu}}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright_{\mu}}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \triangleright_{\mu} \\ & \text{and whenever } \mathcal{P}_{\mathcal{X}} \neq \emptyset, \\ & \text{then } \mathcal{S}_{\pi} \subseteq \triangleright_{\mu} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

Note that the conditions in Theorem 11 are not harmful in practice because the CS problems which are obtained from CS-TRSs normally satisfy those conditions.

*Example 11.* (Continuing Example 10) 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  $\text{Proc}_{\text{subterm}}$  by using the projection  $\pi(\text{IF}) = 3$  and  $\pi(\text{M}) = 1$ :

$$\begin{aligned} \pi(\text{M}(x, y)) &= x \succeq_\mu x = \pi(\text{IF}(\text{gt}(y, 0), \text{minus}(\rho(x), \rho(y)), x)) \\ \pi(\text{IF}(\text{false}, x, y)) &= y \succeq_\mu y = \pi(y) \\ \pi(\text{minus}(\rho(x), \rho(y))) &= \text{minus}(\rho(x), \rho(y)) \triangleright_\mu \rho(x) = \pi(\text{M}(\rho(x), \rho(y))) \end{aligned}$$

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

A variant of the subterm criterion which is based on considering the frozen positions only was also presented in [2]. We extend it now to CS problems. In contrast to the subterm processor, arbitrary (stable) quasi-orderings can be used. As in the subterm processor, we do not need to consider rules in  $\mathcal{R}$ , but only the rules in  $\mathcal{P} \cup \mathcal{S}$ .

**Theorem 12 (Non- $\mu$ -Replacing Projection Processor).** Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem where  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$ . Assume that (1)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , then for all  $s \rightarrow t \in \mathcal{S}_{\sharp}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \mathcal{S}_{\triangleright_\mu} \cup \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\sharp}\}$ . Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded such that

1. for all  $f \in \text{Root}(\mathcal{P})$ ,  $\pi(f) \notin \mu(f)$ ,
2.  $\pi(\mathcal{P}) \subseteq \succsim$ , and,
3. whenever  $\mathcal{S} \neq \emptyset$  and  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , we have that  $\mathcal{S}_{\pi} \subseteq \succsim$

Let  $\mathcal{P}_{\pi, >} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$  and  $\mathcal{S}_{\pi, >} = \{s \rightarrow t \in \mathcal{S}_{\triangleright_\mu} \mid s > t\} \cup \{s \rightarrow t \in \mathcal{S}_{\sharp} \mid s > \pi(t)\}$ . Then, the processor  $\text{Proc}_{\text{NRP}}$  given by

$$\text{Proc}_{\text{NRP}}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, >}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, >}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

*Example 12.* Consider the following TRS  $\mathcal{R}$  [25, Example 1]:

$$\begin{aligned} g(x) &\rightarrow h(x) \\ c &\rightarrow d \\ h(d) &\rightarrow g(c) \end{aligned}$$

together with  $\mu(g) = \mu(h) = \emptyset$ . Note that  $\mathcal{R}$  is  $\mu$ -conservative. Now,  $\text{DP}(\mathcal{R}, \mu)$  consists of the following (noncollapsing) CSDPs:

$$\begin{aligned} G(x) &\rightarrow H(x) \\ H(d) &\rightarrow G(c) \end{aligned}$$

18 Raúl Gutiérrez and Salvador Lucas

and  $\text{unh}(\mathcal{R}, \mu)$  is:

$$c \rightarrow C$$

Then, for the CS problem  $\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$  we can apply  $\text{Proc}_{NRP}(\tau)$  to remove the second pair in  $\tau$  by using the following projection<sup>5</sup>:

$$\begin{aligned}\pi(\mathbf{G}) &= 1 \\ \pi(\mathbf{H}) &= 1\end{aligned}$$

and the following polynomial interpretation:

$$\begin{aligned}[d] &= 1 & [c] &= 0 \\ \pi(\mathbf{G}(x)) &= x \geq x = \pi(\mathbf{H}(x)) \\ \pi(\mathbf{H}(d)) &= d > c = \pi(\mathbf{G}(c))\end{aligned}$$

We can conclude finiteness of the resulting problem using the SCC processor.

## 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 [8].

```
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 [4] can separately prove termination of order-sorted rewriting [21] and CSR, but it is not able to handle programs which combine both of them. Then, we use the transformation developed in [8] to transform this system into a CS-TRS (without sorts). Such a CS-TRS can be found in the

<sup>5</sup> Since  $\text{DP}_{\mathcal{X}}(\mathcal{R}, \mu) = \emptyset$ , we do not need to satisfy (3) in Theorem 12.

Termination Problems Data Base<sup>6</sup> (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<sup>7</sup>.

## 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<sup>8</sup> 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 [12] and VMTL [23] implement the

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

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

context-sensitive dependency pairs using the transformational approach in [1]. The techniques implemented by Jambox [9] 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 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.

<sup>6</sup> <http://www.lri.fr/~marche/tpdb/>

<sup>7</sup> 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>

<sup>8</sup> See <http://termcomp.uibk.ac.at/termcomp/competition/competitionResults.seam?category=10230&competitionId=101722&actionMethod=competition\%2FcategoryList.xhtml\%3AcompetitionCategories.forward\&conversationPropagation=begin>

We have also executed the complete collection of systems of the CSR category<sup>9</sup>, 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 [1], a transformation of collapsing pairs into ‘ordinary’ (i.e., noncollapsing) pairs is introduced by using the new notion of *hiding context* [1, Definition 7]. We easily and naturally included such a transformation as a new processor  $\text{Proc}_{eColl}$  in our framework (see Theorem 4). The claimed advantage of [1] is that the notion of chain is simplified to Item 1 in Definition 5. But, although the definition of chain in [1] is apparently closer to the standard one [13, Definition 3], this does *not* mean that we can use or easily ‘translate’ existing DP-processors (see [13]) to be used with CSR. Besides the narrowing processor in [3, Theorem 16], the reduction pair processor with usable rules in [1, 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 [1] *explicitly* uses them for introducing the new unhiding pairs in [1, 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  $\text{Proc}_{eColl}$  above). Furthermore, the application of such a transformation in the very beginning of the termination analysis of CS-TRSs (as done in [1]) typically leads to obtain a more complex dependency graph (see in Figure 1 (left)) which, as witnessed by our experimental 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 [1] the new improvement in the notion of hiding context (see Definition 2), one has to *redefine* the notion of context-sensitive dependency pair in [1]. 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 [2], handling collapsing pairs is difficult. In [1]

<sup>9</sup> A complete report of our experiments can be found in <http://zenon.dsic.upv.es/muterm/benchmarks/>

this problem is solved by a transformation which disregards collapsing pairs (so we lose 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 [2]. 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 [1]. 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. Alarcón, B., Emmes, F., Fuhs, C., Giesl, J., Gutiérrez, R., Lucas, S., Schneider-Kamp, P., Thiemann, R.: Improving Context-Sensitive Dependency Pairs. In: Cervesato, L., Veith, H., Voronkov, A. (eds.) Proc. of the 15th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'08. LNCS, vol. 5330, pp. 636–651. Springer-Verlag (2008)
2. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. In: Arun-Kumar, S., Garg, N. (eds.) Proc. of the 26th Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06. LNCS, vol. 4337, pp. 297–308. Springer-Verlag (2006)
3. Alarcón, B., Gutiérrez, R., Lucas, S.: Context-Sensitive Dependency Pairs. Information and Computation 208, 922–968 (2010)
4. Alarcón, B., Gutiérrez, R., Lucas, S., Navarro-Marsset, R.: Proving Termination Properties with MU-TERM. In: Proc. of the 13th International Conference on Algebraic Methodology and Software Technology, AMAST'10. LNCS, Springer-Verlag (2010), to appear
5. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Using Dependency Pairs. In: Wolter, F. (ed.) Proc. of the 6th International Symposium on Frontiers of Combining Systems, FroCoS'07. LNCS, vol. 4720, pp. 73–87. Springer-Verlag (2007)
6. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. Theoretical Computer Science 236(1–2), 133–178 (2000)
7. Bruni, R., Meseguer, J.: Semantic Foundations for Generalized Rewrite Theories. Theoretical Computer Science 360(1), 386–414 (2006)
8. Durán, F., Lucas, S., Marché, C., Meseguer, J., Urbain, X.: Proving Operational Termination of Membership Equational Programs. Higher Order Symbolic Computation 21(1-2), 59–88 (2008)
9. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewriting (2009), available at <http://joerg.endrullis.de/jambox.html>
10. Endrullis, J., Hendriks, D.: From Outermost to Context-Sensitive Rewriting. In: Treinen, R. (ed.) Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09. LNCS, vol. 5595, pp. 305–319. Springer-Verlag (2009)

22 Raúl Gutiérrez and Salvador Lucas

11. Fernández, M.L.: Relaxing Monotonicity for Innermost Termination. *Information Processing Letters* 93(3), 117–123 (2005)
12. 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 the 3rd International Joint Conference on Automated Reasoning, IJCAR'06*. LNAI, vol. 4130, pp. 281–286. Springer-Verlag (2006), available at <http://www-i2.informatik.rwth-aachen.de/AProVE>
13. Giesl, J., Thiemann, R., Schneider-Kamp, P., Falke, S.: Mechanizing and Improving Dependency Pairs. *Journal of Automatic Reasoning* 37(3), 155–203 (2006)
14. Gramlich, B., Lucas, S.: Modular Termination of Context-Sensitive Rewriting. In: *Proc. of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming, PPDP'02*. pp. 50–61. ACM Press (2002)
15. Gutiérrez, R., Lucas, S.: Proving Termination in the Context-Sensitive Dependency Pairs Framework. Tech. rep., Universidad Politécnica de Valencia (February 2010), available as Technical Report DSIC-II/02/10
16. Gutiérrez, R., Lucas, S., Urbain, X.: Usable Rules for Context-Sensitive Rewrite Systems. In: Voronkov, A. (ed.) *Proc. of the 19th International Conference on Rewriting Techniques and Applications, RTA'08*. LNCS, vol. 5117, pp. 126–141. Springer-Verlag (2008)
17. Hirokawa, N., Middeldorp, A.: Dependency Pairs Revisited. In: Oostrom, V.v. (ed.) *Proc. of the 15th International Conference on Rewriting Techniques and Applications, RTA'04*. LNCS, vol. 3091, pp. 249–268. Springer-Verlag (2004)
18. Hirokawa, N., Middeldorp, A.: Automating the Dependency Pair Method. *Information and Computation* 199, 172–199 (2005)
19. Lucas, S.: Context-Sensitive Computations in Functional and Functional Logic Programs. *Journal of Functional and Logic Programming* 1998(1), 1–61 (1998)
20. Lucas, S.: Polynomials over the Reals in Proofs of Termination: from Theory to Practice. *RAIRO Theoretical Informatics and Applications* 39(3), 547–586 (2005)
21. Lucas, S., Meseguer, J.: Order-Sorted Dependency Pairs. In: Antoy, S., Albert, E. (eds.) *Proc. of the 10th International ACM SIGPLAN Symposium on Principles and Practice of Declarative Programming, PPDP'08*. pp. 108–119. ACM Press (2008)
22. Ohlebusch, E.: *Advanced Topics in Term Rewriting*. Springer-Verlag (2002)
23. Schernhammer, F., Gramlich, B.: VMTL - A Modular Termination Laboratory. In: Treinen, R. (ed.) *Proc. of the 20th International Conference on Rewriting Techniques and Applications, RTA'09*. LNCS, vol. 5595, pp. 285–294. Springer-Verlag (2009)
24. Thiemann, R., Giesl, J., Schneider-Kamp, P.: Improved Modular Termination Proofs Using Dependency Pairs. In: Basin, D., Rusinowitch, M. (eds.) *Proc. of the 2nd International Joint Conference on Automated Reasoning, IJCAR'04*. LNAI, vol. 3097, pp. 75–90. Springer-Verlag (2004)
25. Zantema, H.: Termination of Context-Sensitive Rewriting. In: Comon, H. (ed.) *Proc. of the 7th International Conference on Rewriting Techniques and Applications, RTA'97*. LNCS, vol. 1232, pp. 172–186. Springer-Verlag (1997)

## A Proof of Theorem 1

The result of this proof can be extracted from [1]. On it, we start from the following definition:

**Definition 17 (Hiding Property [1]).** *A term  $u$  has the hiding property iff*

- $u \in \mathcal{M}_{\infty, \mu}$  and
- whenever  $u \triangleright_{\mu} s \triangleright_{\mu} t'$  for some terms  $s$  and  $t'$  with  $t' \in \mathcal{M}_{\infty, \mu}$ , then  $t'$  is an instance of a hidden term and  $s = C[t']$  for some hiding context  $C[\ ]$ .

**Lemma 1 (Hiding Property Lemma [1]).** *Let  $u$  be a term with the hiding property and let  $u \xrightarrow{\mathcal{R}, \mu} v \triangleright_{\mu} w$  with  $w \in \mathcal{M}_{\infty, \mu}$ . Then  $w$  also has the hiding property.*

*Proof ([1]).* Let  $w \triangleright_{\mu} s \triangleright_{\mu} t'$  for some terms  $s$  and  $t'$  with  $t' \in \mathcal{M}_{\infty, \mu}$ . Clearly, this also implies  $v \triangleright_{\mu} s$ . If already  $u \triangleright s$ , then we must have  $u \triangleright_{\mu} s$  due to the minimality of  $u$ . Thus,  $t'$  is an instance of a hidden term and  $s' = C[t']$  for a hiding context  $C[\ ]$ , since  $u$  has the hiding property. Otherwise,  $u \not\triangleright s$ . There must be a rule  $\ell \rightarrow r \in \mathcal{R}$ , an active context  $D[\ ]$  (i.e., a context where the hole is at an active position), and a substitution  $\delta$  such that  $u = D[\delta(\ell)]$  and  $v = D[\delta(r)]$ . Clearly,  $u \not\triangleright s$  implies  $\delta(\ell) \not\triangleright s$  and  $D[\ ] \not\triangleright s$ . Hence,  $v \triangleright_{\mu} s$  means  $\delta(r) \triangleright_{\mu} s$ . (The root of  $s$  cannot be above  $\square$  in  $D[\ ]$  since those positions would be active.) Note that  $s$  cannot be at or below a variable position of  $r$ , because this would imply  $\delta(\ell) \triangleright s$ . Thus,  $s$  is an instance of a non-variable subterm of  $r$  that is at an inactive position. So there is a  $r' \notin \mathcal{X}$  with  $r \triangleright_{\mu} r'$  and  $s = \delta(r')$ . Recall that  $s \triangleright_{\mu} t'$ , i.e., there is a  $p \in \text{Pos}^{\mu}(s)$  with  $s|_p = t'$ . If  $p$  is a non-variable position of  $r'$ , then  $\delta(r'|_p) = t'$  and  $r'|_p$  is a subterm with defined root at an active position (since  $t' \in \mathcal{M}_{\infty, \mu}$  implies  $\text{root}(t') \in \mathcal{D}$ ). Hence,  $r'|_p$  is a hidden term and thus,  $t'$  is an instance of a hidden term. Moreover, any instance of the context  $C'[\ ] = r'[\ ]|_p$  is hiding. So if we define  $C[\ ]$  to be  $\delta(C'[\ ])$ , then  $s = \delta(r') = \delta(r')[t']_p = \delta(C')[t'] = C[t']$  for the hiding context  $C[\ ]$ . On the contrary, if  $p$  is not a non-variable position of  $r'$ , then  $p = p_1 p_2$  where  $r'|_{p_1}$  is a variable  $x$ . Now  $t'$  is an active subterm of  $\delta(x)$  (more precisely,  $\delta(x)|_{p_2} = t'$ ). Since  $x$  also occurs in  $\ell$ , we have  $\delta(\ell) \triangleright \delta(x)$  and thus  $u \triangleright \delta(x)$ . Due to the minimality of  $u$  this implies  $u \triangleright_{\mu} \delta(x)$  and  $x \in (\text{Var}^{\mu}(\ell) \cap \text{Var}^{\mu}(r)) \setminus (\text{Var}^{\mu}(\ell) \cup \text{Var}^{\mu}(r))$ . Since  $u \triangleright_{\mu} \delta(x) \triangleright_{\mu} t'$ , the hiding property of  $u$  implies that  $t'$  is an instance of a hidden term and that  $\delta(x) = \bar{C}[t']$  for a hiding context  $\bar{C}[\ ]$ . Note that since  $r'|_{p_1}$  is a variable, the context  $C'[\ ]$  around this variable is also hiding (i.e.,  $C' = r'[\ ]|_{p_1}$ ). Thus, the context  $C[\ ] = \delta(C')[\bar{C}[\ ]]$  is hiding as well and  $s = \delta(r') = \delta(r')[\delta(x)[t']_{p_2}]_{p_1} = \delta(C')[\bar{C}[t']] = C[t']$ .

**Proposition 1 (Minimal Root Step[2]).** *Let  $\mathcal{R} = (C \uplus \mathcal{D}, R)$  be a TRS and  $\mu \in M_{\mathcal{R}}$ . Then for all  $t \in \mathcal{M}_{\infty, \mu}$ , there exist  $\ell \rightarrow r \in R$ , a substitution  $\sigma$  and a term  $u \in \mathcal{M}_{\infty, \mu}$  such that  $t \xrightarrow{\geq \Lambda_{\mathcal{R}, \mu}^*} \sigma(\ell) \xrightarrow{\Lambda} \sigma(r) \triangleright_{\mu} u$  and either (1) there is a nonvariable  $\mu$ -replacing subterm  $s$  of  $r$  such that  $u = \sigma(s)$ , or (2) there is  $x \in \text{Var}^{\mu}(r) \setminus \text{Var}^{\mu}(\ell)$  such that  $\sigma(x) \triangleright_{\mu} u$ .*

24 Raúl Gutiérrez and Salvador Lucas

**Theorem 13 (Minimal Sequence in [3]).** *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*

$$t = t_0 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_1(\ell_1) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_2(\ell_2) \xrightarrow{A}_{\mathcal{R}, \mu} \dots$$

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

1.  $t_i = \sigma_i(s_i)$  for some nonvariable term  $s_i$  such that  $r_i \triangleright_{\mu} s_i$ , or
2.  $\sigma_i(x_i) \triangleright_{\mu} t_i$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$  and  $t_i = \theta_i(t'_i)$  for some  $t'_i \in \mathcal{NHT}$  and substitution  $\theta_i$ .

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

$$t = t_0 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_1(\ell_1) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_2(\ell_2) \xrightarrow{A}_{\mathcal{R}, \mu} \dots$$

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

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

*Proof.* Since  $\mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ , by Theorem 13 we have a sequence of the form

$$t = t_0 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_1(\ell_1) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_2(\ell_2) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \dots$$

where, for all  $i \geq 1$ ,  $\ell_i \rightarrow r_i \in R$ ,  $\sigma_i$  are substitutions,  $t_i \in \mathcal{M}_{\infty, \mu}$ , and, by Proposition 1, either (1)  $t_i = \sigma_i(s_i)$  for some  $s_i$  such that  $r_i \triangleright_{\mu} s_i$  or (2)  $\sigma_i(x_i) \triangleright_{\mu} t_i$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$  (and hence  $\sigma(\ell_i) \triangleright_{\mu} t_i$  and  $\sigma(r_i) \triangleright_{\mu} t_i$  as well). If  $\sigma_i(x_i) \triangleright_{\mu} t_i$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$ , it means that  $\sigma(\ell_i) \triangleright_{\mu} C_i[t_i]$ . Since  $t \in \mathcal{T}_{\infty, \mu}$ , it has the hiding property and, by Lemma 1, all  $\sigma(\ell_i)$  satisfies the hiding property. Hence,  $C_i[t_i] = \theta_i(C'_i[t'_i])$  where  $t'_i \in \mathcal{DHT}(\mathcal{R}, \mu)$  and  $C'_i[\ ]$  is a hiding context.

## B Proof of Theorem 2

**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  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\#})$ -chain.*

*Proof. Soundness.*

By contradiction. If  $\mathcal{R}$  is not  $\mu$ -terminating, then there is  $t \in \mathcal{T}_{\infty, \mu}$ . By Theorem 1, there are rules  $\ell_i \rightarrow r_i \in \mathcal{R}$ , matching substitutions  $\sigma_i$ , and terms  $t_i \in \mathcal{M}_{\infty, \mu}$ , for  $i \geq 1$  such that

$$t = t_0 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_1(\ell_1) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_1(r_1) \triangleright_{\mu} t_1 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \sigma_2(\ell_2) \xrightarrow{A}_{\mathcal{R}, \mu} \sigma_2(r_2) \triangleright_{\mu} t_2 \xrightarrow{\geq^A_*}_{\mathcal{R}, \mu} \dots$$

where either (D1)  $t_i = \sigma_i(s_i)$  for some  $s_i$  such that  $r_i \triangleright_{\mu} s_i$  or (D2)  $\sigma_i(x_i) = C_i[t_i]$  for some  $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$  and  $C_i[t_i] = \theta_i(C'_i[t'_i])$  for some  $t'_i \in \mathcal{NHT}$  and hiding context  $C'_i[\square]$ . Furthermore, since  $t_{i-1} \xrightarrow{\mathcal{R}, \mu}^{\Lambda} \sigma_i(\ell_i)$  and  $t_{i-1} \in \mathcal{M}_{\infty, \mu}$  (in particular,  $t_0 = t \in \mathcal{T}_{\infty, \mu} \subseteq \mathcal{M}_{\infty, \mu}$ ),  $\sigma_i(\ell_i) \in \mathcal{M}_{\infty, \mu}$  for all  $i \geq 1$ . Note that, since  $t_i \in \mathcal{M}_{\infty, \mu}$ , we have that  $t_i^{\sharp}$  is  $\mu$ -terminating (with respect to  $\mathcal{R}$ ), because all  $\mu$ -replacing subterms of  $t_i$  (hence of  $t_i^{\sharp}$  as well) are  $\mu$ -terminating and  $\text{root}(t_i^{\sharp})$  is not a defined symbol of  $\mathcal{R}$ .

First, note that  $\text{DP}(\mathcal{R}, \mu)$  is a TRS  $\mathcal{P}$  over the signature  $\mathcal{G} = \mathcal{F} \cup \mathcal{D}^{\sharp}$  and  $\mu^{\sharp} \in M_{\mathcal{F} \cup \mathcal{G}}$  as required by Definition 5. Furthermore,  $\mathcal{P}_{\mathcal{G}} = \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$  and  $\mathcal{P}_{\mathcal{X}} = \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . We can define an infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$ -chain using CSDPs  $u_i \rightarrow v_i$  for  $i \geq 1$ , where  $u_i = \ell_i^{\sharp}$  and

1.  $v_i = s_i^{\sharp}$  if (D1) holds. Since  $t_i \in \mathcal{M}_{\infty, \mu}$ , we have that  $\text{root}(s_i) \in \mathcal{D}$  and, because  $t_i = \sigma_i(s_i)$  and  $\sigma_i(s_i) \hookrightarrow^* \sigma_{i+1}(\ell_{i+1})$ ,  $\text{REN}^{\mu}(s_i)$  is  $\mu$ -narrowable [3, Proposition 5]. Furthermore, if we assume that  $s_i$  is a  $\mu$ -replacing subterm of  $\ell_i$  (i.e.,  $\ell_i \triangleright_{\mu} s_i$ ), then  $\sigma_i(\ell_i) \triangleright_{\mu} \sigma_i(s_i)$  which (since  $\sigma_i(s_i) = t_i \in \mathcal{M}_{\infty, \mu}$ ) contradicts that  $\sigma_i(\ell_i) \in \mathcal{M}_{\infty, \mu}$ . Thus,  $\ell_i \not\triangleright_{\mu} s_i$ . Hence,  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ . Furthermore,  $t_i^{\sharp} = \sigma_i(v_i)$  is  $\mu$ -terminating. Finally, since  $t_i = \sigma_i(s_i) \xrightarrow{\mathcal{R}, \mu}^{\Lambda} \sigma_{i+1}(\ell_{i+1})$  and  $\mu^{\sharp}$  extends  $\mu$  to  $\mathcal{F} \cup \mathcal{D}^{\sharp}$  by  $\mu^{\sharp}(f^{\sharp}) = \mu(f)$  for all  $f \in \mathcal{D}$ , we also have that  $\sigma_i(v_i) = \sigma_i(s_i^{\sharp}) \hookrightarrow_{\mathcal{R}, \mu^{\sharp}}^* \sigma_{i+1}(u_{i+1})$ .
2.  $v_i = x_i$  if (D2) holds. Clearly,  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ . As discussed above,  $t_i^{\sharp}$  is  $\mu$ -terminating. Since  $\sigma_i(x_i) = C_i[t_i]$ , we have that  $\sigma_i(v_i) = C_i[t_i]$ . By the hiding property, we know that  $C_i[\square]$  is an instance of hiding context  $C'_i[\square]$ , then we have that  $\theta_i(C'_i)[t_i] \xrightarrow{\mathcal{S}_{\triangleright_{\mu}, \mu}}^{\Lambda} t_i$ . And we also know that  $t_i$  is an instance  $\theta_i(t'_i)$  of a hidden term  $t'_i \in \mathcal{NHT}(\mathcal{R}, \mu)$ . Thus  $t'_i \rightarrow t_i^{\sharp} \in \mathcal{S}_{\sharp}$  and we have  $\theta(t'_i) \xrightarrow{\mathcal{S}_{\triangleright_{\mu}, \mu}}^{\Lambda} \theta(t_i^{\sharp})$ . Finally, since  $t_i \xrightarrow{\mathcal{R}, \mu}^{\Lambda} \sigma_{i+1}(\ell_{i+1})$ , again we have that  $u_i^{\sharp} \hookrightarrow_{\mathcal{R}, \mu^{\sharp}}^* \sigma_{i+1}(u_{i+1})$ .

Regarding  $\sigma$ , w.l.o.g. we can assume that  $\text{Var}(\ell_i) \cap \text{Var}(\ell_j) = \emptyset$  for all  $i \neq j$ , and therefore  $\text{Var}(u_i) \cap \text{Var}(u_j) = \emptyset$  as well. Then,  $\sigma$  is given by  $\sigma(x) = \sigma_i(x)$  whenever  $x \in \text{Var}(u_i)$  for  $i \geq 1$ . From the discussion in points (1) and (2) above, we conclude that the CSDPs  $u_i \rightarrow v_i$  for  $i \geq 1$  together with  $\sigma$  define an infinite minimal  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$ -chain which contradicts our initial assumption.

#### Completeness.

By contradiction. If there is an infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^{\sharp})$ -chain, then there is a substitution  $\sigma$  and dependency pairs  $u_i \rightarrow v_i \in \text{DP}(\mathcal{R}, \mu)$  such that

1.  $\sigma(v_i) \hookrightarrow_{\mathcal{R}, \mu^{\sharp}}^* \sigma(u_{i+1})$ , if  $u_i \rightarrow v_i \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , and
2. if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , then  $\sigma(v_i) \xrightarrow{\mathcal{S}_{\triangleright_{\mu}, \mu}}^{\Lambda} \sigma(u_{i+1}) \circ \xrightarrow{\mathcal{S}_{\triangleright_{\mu}, \mu}}^{\Lambda} \sigma(x_i) \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$ .

for  $i \geq 1$ . Now, consider the first dependency pair  $u_1 \rightarrow v_1$  in the sequence:

26 Raúl Gutiérrez and Salvador Lucas

1. If  $u_1 \rightarrow v_1 \in \text{DP}_{\mathcal{F}}(\mathcal{R}, \mu)$ , then  $v_1^\sharp$  is a  $\mu$ -replacing subterm of the right-hand-side  $r_1$  of a rule  $l_1 \rightarrow r_1$  in  $\mathcal{R}$ . Therefore,  $r_1 = C_1[v_1^\sharp]_{p_1}$  for some  $p_1 \in \text{Pos}^\mu(r_1)$  and we can perform the  $\mu$ -rewriting step  $t_1 = \sigma(u_1) \hookrightarrow_{\mathcal{R}, \mu} \sigma(r_1) = \sigma(C_1[\sigma(v_1^\sharp)]_{p_1}) = s_1$ , where  $\sigma(v_1^\sharp) = \sigma(v_1) \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_2)$  and  $\sigma(u_2)$  initiates an infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ -chain. Note that  $p_1 \in \text{Pos}^\mu(s_1)$ .
2. If  $u_1 \rightarrow x \in \text{DP}_{\mathcal{X}}(\mathcal{R}, \mu)$ , then there is a rule  $l_1 \rightarrow r_1$  in  $\mathcal{R}$  such that  $u_1 = l_1^\sharp$ , and  $x \in \text{Var}^\mu(r_1) \setminus \text{Var}^\mu(l_1)$ , i.e.,  $r_1 = C_1[x]_{q_1}$  for some  $q_1 \in \text{Pos}^\mu(r_1)$ . Furthermore, if  $\sigma(v_1) = C_1[t_1] \xrightarrow{A}_{\mathcal{S}_{\triangleright \mu}, \mu} t_1$  this means that  $C_1[\square]$  is an instance of a hiding context  $C_1'[\square]$ ,  $C_1[\square] = \theta_1(C_1')[\square]$ . Furthermore,  $t_1$  is  $\mu$ -replacing in  $C_1[t_1]$ . If  $t_1 \xrightarrow{A}_{\mathcal{S}_{\triangleright \mu}, \mu} s_1$  means that  $t_1 = \theta_1(t_1')$  for some  $t_1' \in \mathcal{DHT}$ , then  $s_1 \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_2)$  and  $\sigma(u_2)$  initiates an infinite  $(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\sharp)$ -chain. Note that  $p_1 = q_1 \cdot p_1' \in \text{Pos}^\mu(s_1)$  where  $p_1'$  is the position of the hole in  $C_1[\square]_{p_1'}$ .

Since  $\mu^\sharp(f^\sharp) = \mu(f)$ , and  $p_1 \in \text{Pos}^\mu(s_1)$ , we have that  $s_1 \hookrightarrow_{\mathcal{R}, \mu}^* t_2[\sigma(u_2)]_{p_1} = t_2$  and  $p_1 \in \text{Pos}^\mu(t_2)$ . Therefore, we can build in that way an infinite  $\mu$ -rewrite sequence

$$t_1 \hookrightarrow_{\mathcal{R}, \mu} s_1 \hookrightarrow_{\mathcal{R}, \mu}^* t_2 \hookrightarrow_{\mathcal{R}, \mu} \dots$$

which contradicts the  $\mu$ -termination of  $\mathcal{R}$ .

### C Proof of Theorem 4

In the following proofs, the notion  $\left\{ \begin{array}{c} \rightarrow_1 \\ \rightarrow_2 \end{array} \right\}$  for reduction relations  $\rightarrow_1$  and  $\rightarrow_2$  means that either  $s \rightarrow_1 t$  or  $s \rightarrow_2 t$ , that is,  $\left\{ \begin{array}{c} \rightarrow_1 \\ \rightarrow_2 \end{array} \right\} = \rightarrow_1 \cup \rightarrow_2$ .

**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  $P_{\mathcal{U}}$  be given by the following rules:*

- $u \rightarrow \mathcal{U}(x)$  for every  $u \rightarrow x \in \mathcal{P}_{\mathcal{X}}$ ,
- $\mathcal{U}(s) \rightarrow \mathcal{U}(t)$  for every  $s \rightarrow t \in \mathcal{S}_{\triangleright \mu}$ , and
- $\mathcal{U}(s) \rightarrow t$  for every  $s \rightarrow t \in \mathcal{S}_{\sharp}$ .

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

*Proof. Soundness.* We prove first that the existence of an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain implies the existence of an infinite minimal  $(\mathcal{P}', \mathcal{R}, \emptyset, \mu')$ -chain.

First, note that  $\mathcal{P}'$  is well-defined as a TRS. Consider an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$ :

$$\sigma(u_1) \left\{ \begin{array}{c} \hookrightarrow_{\mathcal{P}_{\mathcal{G}, \mu}'} \\ \hookrightarrow_{\mathcal{P}_{\mathcal{X}, \mu}} \circ \xrightarrow{A}_{\mathcal{S}_{\triangleright \mu}, \mu} \circ \xrightarrow{A}_{\mathcal{S}_{\sharp}, \mu} \end{array} \right\} t_1 \hookrightarrow_{\mathcal{R}, \mu}^* \sigma(u_2) \left\{ \begin{array}{c} \hookrightarrow_{\mathcal{P}_{\mathcal{G}, \mu}'} \\ \hookrightarrow_{\mathcal{P}_{\mathcal{X}, \mu}} \circ \xrightarrow{A}_{\mathcal{S}_{\triangleright \mu}, \mu} \circ \xrightarrow{A}_{\mathcal{S}_{\sharp}, \mu} \end{array} \right\} \dots$$

for some substitution  $\sigma$ , where, for all  $i \geq 1$ ,  $t_i$  is  $\mu$ -terminating and, (1) if  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$  and the chain is maintained unchanged (2) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{P}_{\mathcal{X}}$ , then  $\sigma(v_i) \xrightarrow{\Lambda^*_{\mathcal{S}_{\triangleright\mu}, \mu}} s_i \xrightarrow{\Lambda_{\mathcal{S}_{\triangleright\mu}, \mu}} t_i$ . Note that, instead of  $u_i \rightarrow x_i$ , we have  $u_i \rightarrow \mathbf{U}(x_i)$ ; for all rules  $s \rightarrow t$  such that  $s \triangleright_{\mu} t$  ( $s \rightarrow t \in \mathcal{S}_{\triangleright\mu}$ ) we have  $\mathbf{U}(s) \rightarrow \mathbf{U}(t) \in \mathcal{P}'$ ; and for all rules  $s \rightarrow t \in \mathcal{S}_{\sharp}$  we have  $\mathbf{U}(s) \rightarrow t \in \mathcal{P}'$ . Therefore, for all terms  $s, t$ ,  $s \xrightarrow{\Lambda^*_{\mathcal{S}_{\triangleright\mu}, \mu}} t$  if and only if  $\mathbf{U}(s) \xrightarrow{\Lambda^*_{\mathcal{P}', \mu}} \mathbf{U}(t)$  (where we only use the rules in  $\mathcal{P}'$  which are of the form  $\mathbf{U}(s) \rightarrow \mathbf{U}(t)$ ), and  $s \xrightarrow{\Lambda_{\mathcal{S}_{\triangleright\mu}, \mu}} t$  if and only if  $\mathbf{U}(s) \xrightarrow{\Lambda_{\mathcal{P}', \mu}} t$ . Hence, we obtain:

$$\begin{array}{l} \sigma(u_i) \\ \xrightarrow{\Lambda_{\mathcal{P}', \mu'}} \mathbf{U}(\sigma(x_i)) \text{ a collapsing pair is applied} \\ \xrightarrow{\Lambda^*_{\mathcal{P}', \mu'}} \mathbf{U}(s_i) \quad \text{we apply } \mathcal{S}_{\triangleright\mu}\text{-rules} \\ \xrightarrow{\Lambda_{\mathcal{P}', \mu'}} t_i \quad \text{we apply a } \mathcal{S}_{\sharp}\text{-rule} \end{array}$$

Thus, we obtain an infinite minimal  $(\mathcal{P}', \mathcal{R}, \emptyset, \mu')$ -chain, as desired. In particular, we note that all steps with  $\mathcal{P}_{\mathbf{U}}$  are performed at the root, we do not require any reduction below symbol  $\mathbf{U}$ , hence  $\mu'(\mathbf{U}) = \emptyset$  is enough to perform them.

**Completeness.** By contradiction. If there is an infinite  $(\mathcal{P}', \mathcal{R}, \emptyset, \mu')$ -chain, then there is a substitution  $\sigma$  and pairs  $u_i \rightarrow v_i \in \mathcal{P}'$  such that

1. If  $u_i \rightarrow v_i \in \mathcal{P}' \setminus \mathcal{P}_{\mathbf{U}}$  and  $\sigma(v_i) \xrightarrow{\Lambda^*_{\mathcal{R}, \mu}} \sigma(u_{i+1})$ , then  $u_i \rightarrow v_i \in \mathcal{P}$  and  $\sigma(v_i) \xrightarrow{\Lambda^*_{\mathcal{R}, \mu}} \sigma(u_{i+1})$ ,
2. if  $u_i \rightarrow v_i = u_i \rightarrow \mathbf{U}(x_i) \in \mathcal{P}'$  and  $\sigma(u_i) \xrightarrow{\Lambda^*_{\mathcal{R}, \mu}} \sigma(v_{i+1})$  where  $\sigma(v_{i+1}) = \mathbf{U}(v'_{i+1})$ , then there is a pair  $u_i \rightarrow x_i \in \mathcal{P}$  such that  $\sigma(x_i) = v'_{i+1}$ ,
3. if  $u_i \rightarrow v_i = \mathbf{U}(s) \rightarrow \mathbf{U}(t) \in \mathcal{P}'$  then there is a pair in  $\mathcal{S}_{\triangleright\mu}$ :  $s \rightarrow t \in \mathcal{S}_{\triangleright\mu}$ , and
4. if  $u_i \rightarrow v_i = \mathbf{U}(s) \rightarrow t \in \mathcal{P}'$  then there is a pair  $s \rightarrow t \in \mathcal{S}_{\sharp}$ .

Hence, we can build in that way an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain which contradicts the  $\mu$ -termination of  $\mathcal{R}$ .

## D Proof of Theorem 5

**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  $s \rightarrow t \in \mathcal{S}_{\sharp}$ ,  $t \notin \mathcal{X}$  (2)  $\{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_{\sharp}\} \cap \mathcal{D} = \emptyset$  and (3) for all  $s \rightarrow t \in \mathcal{S}_{\sharp}$ , we have that  $s = \mathbf{f}(s_1, \dots, s_k)$  and  $t = \mathbf{g}(s_1, \dots, s_k)$  for some  $k \in \mathbb{N}$ , function symbols  $\mathbf{f}, \mathbf{g} \in \mathcal{H}$ , and terms  $s_1, \dots, s_k$ . Then, the processors  $\text{Proc}_{\text{Coll}}$  given by*

$$\text{Proc}_{\text{Coll}}(\tau) = \begin{cases} \emptyset & \text{if } \mathcal{P} = \mathcal{P}_{\mathcal{X}}^{\lambda} \text{ and} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

*is sound and complete.*

28 Raúl Gutiérrez and Salvador Lucas

*Proof.* If  $\mathcal{P} = \mathcal{P}_{\mathcal{X}}^1$ , then we proceed by contradiction. Assume that there is an infinite chain which only uses dependency pairs  $u_i \rightarrow x_i \in \mathcal{P}_{\mathcal{X}}^1$  for all  $i \geq 1$ . Let  $f_i = \text{root}(u_i)$  for  $i \geq 1$ . Then, by definition of  $\mathcal{P}_{\mathcal{X}}^1$ , for all  $i \geq 1$  there is  $j_i \in \{1, \dots, \text{ar}(f_i)\} \setminus \mu(f_i)$  such that  $u_i|_{j_i} \triangleright x_i$ . We have that  $\sigma(u_i)|_{j_i} \triangleright \sigma(x_i) \xrightarrow{A}^*_{\mathcal{S}_{\mathcal{D}, \mu}, \mu} s_i \xrightarrow{A}_{\mathcal{S}_{\mathcal{D}, \mu}, \mu} t_i$  for some terms  $s_i, t_i$  such that  $t_i \xrightarrow{*}_{\mathcal{R}, \mu} \sigma(u_{i+1})$ . Since  $s_i = \theta_i(s'_i)$  and  $t_i = \theta_i(t'_i)$  for some  $s'_i \rightarrow t'_i \in \mathcal{S}_{\#}$  and substitution  $\theta_i$ , by (1), we have that  $\text{root}(t_i) = \text{root}(t'_i)$ . By (2), we have that  $\text{root}(t_i) \notin \mathcal{D}$ , i.e.,  $t_i$  cannot be  $\mu$ -rewritten at the root by using the rules in  $\mathcal{R}$ . Hence,  $\text{root}(t_i) = \text{root}(u_{i+1}) = f_{i+1}$  and  $j_{i+1} \notin \mu(f_{i+1})$ . Note that  $\sigma(x_i) \triangleright_{\mu} s_i$ . Since  $t_i \xrightarrow{A}_{\mathcal{R}, \mu}^* \sigma(u_{i+1})$  and no  $\mu$ -rewriting is possible on the  $j_{i+1}$ -th immediate subterm  $t_i|_{j_{i+1}}$  of  $t_i$  and also  $s_i$  and  $t_i$  only differ in the root symbol (due to (3)), it follows that  $\sigma(u_i)|_{j_i} \triangleright s_i$ ,  $s_i|_{j_i} = t_i|_{j_i}$  (due to (3)) and  $t_i|_{j_i} = \sigma(u_{i+1})|_{j_{i+1}} \triangleright \sigma(x_{i+1})$ , i.e.,  $\sigma(x_i) \triangleright \sigma(x_{i+1})$  for all  $i \geq 1$ . We get an infinite sequence  $\sigma(x_1) \triangleright \sigma(x_2) \triangleright \dots$  which contradicts well-foundedness of  $\triangleright$ .

## E Proof of Theorem 6

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

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

(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.

*Proof.* We prove soundness by contradiction. Assume that  $\text{Proc}_{SCC}$  is not sound. Then, there is a CS problem  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  such that, for all  $\tau' \in \text{Proc}_{SCC}(\tau)$ ,  $\tau'$  is finite but  $\tau$  is not finite. Thus, there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$ . Since  $\mathcal{P}$  contains a finite number of pairs, there is  $\mathcal{P}' \subseteq \mathcal{P}$  and a tail  $B$  of  $A$ , which is an infinite minimal  $(\mathcal{P}', \mathcal{R}, \mathcal{S}, \mu)$ -chain where all pairs in  $\mathcal{P}'$  are infinitely often used. According to Definition 7, this means that  $\mathcal{P}'$  is a cycle in  $\mathbf{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ . Hence  $\mathcal{P}'$  belongs to some SCC with nodes in  $\mathcal{Q}$ , i.e.,  $\mathcal{P}' \subseteq \mathcal{Q}$ . Thus,  $B$  is an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu)$ -chain, i.e.,  $\tau' = (\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu)$  is not finite. Since  $\tau' \in \text{Proc}_{SCC}(\tau)$ , we obtain a contradiction.

With regard to completeness, since  $\mathcal{Q} \subseteq \mathcal{P}$  for some SCC in  $\mathbf{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  with nodes in  $\mathcal{Q}$ , every infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mathcal{S}, \mu)$ -chain is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain. Hence, the processor is complete as well.

## F Proof of Theorem 7

We need some previous results to prove Theorem 7.

**Proposition 2 (TCAP $_{\mathcal{R}}^{\mu}$  Property [3]).** *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mathcal{G}$  be a signature and  $\mu \in M_{\mathcal{F} \cup \mathcal{G}}$ . Let  $t, u \in \mathcal{T}(\mathcal{F} \cup \mathcal{G}, \mathcal{X})$  be such that  $\text{Var}(t) \cap \text{Var}(u) = \emptyset$ . If there are substitutions  $\theta$  and  $\theta'$  such that  $\theta(t) \xrightarrow{*}_{\mathcal{R}, \mu} \theta'(u)$ , then  $\text{TCAP}_{\mathcal{R}}^{\mu}(t)$  and  $u$  unify.*

**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  $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  contains the CS graph  $\mathcal{G}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ .*

*Proof.* Direct as a consequence of Definition 5, Definition 7 and Proposition 2.

## G Proof of Theorem 8

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

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

where

- $\mathcal{S}_{\mathcal{Q}} = \emptyset$  if  $\mathcal{Q}_{\mathcal{X}} = \emptyset$ .
- $\mathcal{S}_{\mathcal{Q}} = \mathcal{S}_{\mathcal{D}_{\mu}} \cup \{s \rightarrow t \mid s \rightarrow t \in \mathcal{S}_{\mathcal{D}_{\mu}}, \text{TCAP}_{\mathcal{R}}^{\mu}(t) \text{ and } u' \text{ unify}\}$  for some  $u' \rightarrow v' \in \mathcal{Q}$  if  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ .

is sound and complete.

*Proof.* Direct as a consequence of Theorem 6, Definition 9 and Theorem 7.

## H Proof of Theorem 9

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

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

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

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

is sound and complete.

*Proof.* Completeness is obvious, since  $\mathcal{P} \setminus \mathcal{P}_{\sqsupset} \subseteq \mathcal{P}$ . Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$ , but there is no infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{S} \setminus \mathcal{S}_{\sqsupset}, \mathcal{R}, \mu)$ -chain. Due to the finiteness of  $\mathcal{P}$  and  $\mathcal{S}$ , we can assume that there are subsets  $\mathcal{Q} \subseteq \mathcal{P}$  and  $\mathcal{T} \subseteq \mathcal{S}$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \left\{ \begin{array}{c} \xrightarrow{\mathcal{Q}_{\mathcal{G}, \mu}} \\ \xrightarrow{\mathcal{Q}_{\mathcal{X}, \mu}} \circ \xrightarrow{\Lambda_{\mathcal{D}_{\mu}, \mu}} \circ \xrightarrow{\Lambda_{\mathcal{T}_2, \mu}} \end{array} \right\} t_1 \xrightarrow{\mathcal{R}, \mu} \sigma(u_2) \left\{ \begin{array}{c} \xrightarrow{\mathcal{Q}_{\mathcal{G}, \mu}} \\ \xrightarrow{\mathcal{Q}_{\mathcal{X}, \mu}} \circ \xrightarrow{\Lambda_{\mathcal{D}_{\mu}, \mu}} \circ \xrightarrow{\Lambda_{\mathcal{T}_2, \mu}} \end{array} \right\} \dots$$

30 Raúl Gutiérrez and Salvador Lucas

for some substitution  $\sigma$ , where all pairs in  $\mathcal{Q}$  and all rules in  $\mathcal{T}$  are infinitely often used (note that, if  $\mathcal{T} \neq \emptyset$ , then  $\mathcal{T}_{\#} \neq \emptyset$  and  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ ), and, for all  $i \geq 1$ , (1) if  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$  and (2) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$ , then  $\sigma(u_i) \hookrightarrow_{\mathcal{Q}_{\mathcal{X}}} \sigma(v_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{T}_{\#}, \mu}^*} \sigma(x_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{T}_{\#}, \mu}^*} t_i$ .

Since  $u_i (\succsim \cup \sqsupset) v_i$  for all  $u_i \rightarrow v_i \in \mathcal{Q} \subseteq \mathcal{P}$ , by stability of  $\succsim$  and  $\sqsupset$ , we have  $\sigma(u_i) (\succsim \cup \sqsupset) \sigma(v_i)$  for all  $i \geq 1$ .

No pair  $u \rightarrow v \in \mathcal{Q}$  satisfies that  $u \sqsupset v$  and no rule  $\ell \rightarrow r \in \mathcal{T}$  satisfies  $s \sqsupset t$ . Otherwise, we get a contradiction by considering the following two cases:

1. If  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{R}, \mu}^*} \sigma(u_{i+1})$  and  $t_i \succsim \sigma(u_{i+1})$ . Since we have  $\sigma(u_i) (\succsim \cup \sqsupset) \sigma(v_i) = t_i$ , by using transitivity of  $\succsim$  and compatibility between  $\succsim$  and  $\sqsupset$ , we conclude that  $\sigma(u_i) (\succsim \cup \sqsupset) \sigma(u_{i+1})$ .
2. If  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$  (which is not empty whenever  $\mathcal{T}$  is not empty), then  $\sigma(v_i) = \sigma(x_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{T}_{\#}, \mu}^*} \sigma(s_i)$ . Since  $\ell_j (\succeq \cup \succsim \cup \sqsupset) r_j$  for all  $\ell_j \rightarrow r_j \in \mathcal{T}_{\#}$ , we have  $\sigma(v_i) = \sigma(x_i) (\succeq \cup \succsim \cup \sqsupset) \sigma(s_i)$ . Furthermore, we are assuming that  $\ell (\succeq \cup \succsim \cup \sqsupset) r$  for all  $\ell \rightarrow r \in \mathcal{T}_{\#}$ . Since  $\sigma(s_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{T}_{\#}}^*} \sigma(t_i)$  for some  $s_i \rightarrow t_i \in \mathcal{T}_{\#}$  and, by stability of the (quasi-)orderings we have that  $\sigma(s_i) (\succeq \cup \succsim \cup \sqsupset) \sigma(t_i)$ . Hence, by transitivity of  $\succsim$  (and compatibility of  $\succsim$ ,  $\sqsupset$  and  $\succeq$ ), we have  $\sigma(v_i) = \sigma(x_i) (\succeq \cup \succsim \cup \sqsupset) \sigma(t_i)$ . Since  $\sigma(t_i) \xrightarrow{\overset{\Delta}{\hookrightarrow}_{\mathcal{R}, \mu}^*} \sigma(u_{i+1})$ , we also have that, for all  $i \geq 1$ ,  $\sigma(t_i) \succsim \sigma(u_{i+1})$ . Therefore, again by transitivity of  $\succsim$  and compatibility of  $\succsim$ ,  $\succeq$  and  $\sqsupset$ , we conclude that  $\sigma(u_i) (\succeq \cup \succsim \cup \sqsupset) t_i \succsim \sigma(u_{i+1})$  and hence  $\sigma(u_i) (\succsim \cup \sqsupset) \sigma(u_{i+1})$ .

Since  $u \rightarrow v$  and  $\ell \rightarrow r$  occur infinitely often in  $B$ , there is an infinite set  $\mathcal{I} \subseteq \mathbb{N}$  of pairs such that  $\sigma(u_i) \sqsupset \sigma(u_{i+1})$  for all  $i \in \mathcal{I}$ . Thus, by using the compatibility conditions of the  $\mu$ -reduction triple, we obtain an infinite decreasing  $\sqsupset$ -sequence which contradicts well-foundedness of  $\sqsupset$ .

Therefore,  $B$  is an infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\sqsupset}, \mu)$ -chain, thus leading to a contradiction.

## I Proof of Theorem 10

In order to prove Theorem 10, we use the following definitions.

**Definition 18 (Basic  $\mu$ -Interpretation [16]).** Let  $(\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mu}$  and  $\Delta \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathfrak{c}\}, \mathcal{X})$  where  $\perp$  is a fresh constant symbol and  $\mathfrak{c}$  is a fresh binary symbol. The basic  $\mu$ -interpretation  $\mathcal{I}_{0, \Delta, \mu}$  is a mapping from  $\mu$ -terminating terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F} \cup \{\perp, \mathfrak{c}\}, \mathcal{X})$  defined as follows:

$$\mathcal{I}_{0, \Delta, \mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ \mathfrak{f}(\mathcal{I}_{0, \Delta, \mu, \mathfrak{f}, 1}(t_1), \dots, \mathcal{I}_{0, \Delta, \mu, \mathfrak{f}, n}(t_k)) & \text{if } t = \mathfrak{f}(t_1, \dots, t_k) \\ & \text{and } \mathfrak{f} \in \Delta \\ \mathfrak{c}(\mathcal{I}_{0, \Delta, \mu, \mathfrak{f}, 1}(t_1), \dots, \mathcal{I}_{0, \Delta, \mu, \mathfrak{f}, n}(t_k)), t' & \text{if } t = \mathfrak{f}(t_1, \dots, t_k) \\ & \text{and } \mathfrak{f} \notin \Delta \end{cases}$$

$$\begin{aligned}
\text{where } \mathcal{I}_{0,\Delta,\mu,t,i}(t) &= \begin{cases} \mathcal{I}_{0,\Delta,\mu}(t) & \text{if } i \in \mu(\mathbf{f}) \\ t & \text{if } i \notin \mu(\mathbf{f}) \end{cases} \\
t' &= \text{order}(\{\mathcal{I}_{0,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\}) \\
\text{order}(T) &= \begin{cases} \perp, & \text{if } T = \emptyset \\ c(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}
\end{aligned}$$

**Lemma 2.** [16] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $t$  in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$ . If  $t$  is  $\mu$ -terminating then  $\mathcal{I}_{0,\Delta,\mu}$  is well-defined.*

*Proof.* According to Definition 18, to obtain an infinite term as result of  $\mathcal{I}_{0,\Delta,\mu}(t)$ , we would have to perform an infinite number of applications of the function  $\text{order}(\{\mathcal{I}_{0,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\})$ . Since  $t$  is  $\mu$ -terminating, then all subterms at  $\mu$ -replacing positions are  $\mu$ -terminating as well. Hence, this is not possible.

**Definition 19.** [16] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\sigma$  be a substitution. Let  $\Delta \subseteq \mathcal{F}$ . We denote by  $\sigma_{\mathcal{I}_{0,\Delta,\mu}}: \mathcal{T}(\mathcal{F}, \mathcal{X}) \rightarrow \mathcal{T}(\mathcal{F}, \mathcal{X})$  a function that, given a term  $t$  replaces occurrences of  $x \in \text{Var}(t)$  at position  $p$  in  $t$  by either  $\mathcal{I}_{0,\Delta,\mu}(\sigma(x))$  if  $p \in \text{Pos}^{\mu}(t)$  and  $\sigma(x)$  is  $\mu$ -terminating, or  $\sigma(x)$  otherwise.*

**Proposition 3.** [16] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\sigma$  be a substitution. Let  $\Delta \subseteq \mathcal{F}$ . Let  $t$  be a term such that  $\text{Var}^{\mu}(t) \cap \text{Var}^{\#}(t) = \emptyset$ . Let  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}$  be a substitution given by*

$$\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(x) = \begin{cases} \mathcal{I}_{0,\Delta,\mu}(\sigma(x)) & \text{if } x \in \text{Var}^{\mu}(t) \text{ and } \sigma(x) \text{ is } \mu\text{-terminating} \\ \sigma(x) & \text{otherwise} \end{cases}$$

*Then,  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t) = \sigma_{\mathcal{I}_{0,\Delta,\mu}}(t)$ .*

*Proof.* By structural induction on  $t$ .

- If  $t = x \in \mathcal{X}$  we have two possibilities: (1) if  $\sigma(x)$  is  $\mu$ -terminating, then  $\sigma_{\mathcal{I}_{0,\Delta,\mu}} = \mathcal{I}_{0,\Delta,\mu}(\sigma(x))$  and  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t} = \mathcal{I}_{0,\Delta,\mu}(\sigma(x))$  (since  $x \in \text{Var}^{\mu}(t)$  and  $\sigma(x)$  is  $\mu$ -terminating), and (2) if  $\sigma(x)$  is non- $\mu$ -terminating, then  $\sigma_{\mathcal{I}_{0,\Delta,\mu}} = \sigma(x)$  and  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t} = \sigma(x)$  (since  $x \in \text{Var}^{\mu}(t)$  and  $\sigma(x)$  is non- $\mu$ -terminating)
- If  $t = \mathbf{f}(t_1, \dots, t_k)$  we have two possibilities: (1) if  $i \in \mu(\mathbf{f})$  we can apply the induction hypothesis to  $t_i$  and obtain that  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t_i) = \sigma_{\mathcal{I}_{0,\Delta,\mu}}(t_i)$ , and (2) if  $t = \mathbf{f}(t_1, \dots, t_k)$  and  $i \notin \mu(\mathbf{f})$  then  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t_i) = \sigma_{\mathcal{I}_{0,\Delta,\mu}}(t_i) = \sigma(t_i)$ .

Since  $\text{Var}^{\mu}(t) \cap \text{Var}^{\#}(t) = \emptyset$  we can ensure that for all variable  $x$  in  $t$ : (1) if  $x \in \text{Var}^{\mu}(t)$  and  $\sigma(x)$  is  $\mu$ -terminating then  $x \notin \text{Var}^{\#}(t)$  and  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t} = \sigma_{\mathcal{I}_{0,\Delta,\mu}} = \mathcal{I}_{0,\Delta,\mu}(\sigma(x))$ , and (2) if  $x \in \text{Var}^{\#}(t)$  (then  $x \notin \text{Var}^{\mu}(t)$ ) or  $\sigma(x)$  is non- $\mu$ -terminating then  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t} = \sigma_{\mathcal{I}_{0,\Delta,\mu}} = \sigma(x)$ .

**Lemma 3.** [16] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\Delta \subseteq \mathcal{F}$ . Let  $t$  be a term and  $\sigma$  be a substitution. If  $\sigma(t)$  is  $\mu$ -terminating, then  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ . If  $t$  only contain  $\Delta$ -symbols at  $\mu$ -replacing positions, then we have  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .*

32 Raúl Gutiérrez and Salvador Lucas

*Proof.* By structural induction on  $t$ :

- If  $t$  is a variable then  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .
- If  $t = f(t_1, \dots, t_k)$  then
  - If  $f \in \Delta$  then  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k)))$ . Terms  $\sigma(t_i)$  are  $\mu$ -terminating for  $i \in \mu(f)$ . By induction hypothesis, for all terms  $t_i$  s.t.  $i \in \mu(f)$ , we have  $\mathcal{I}_{0,\Delta,\mu,f,i}(\sigma(t_i)) = \mathcal{I}_{0,\Delta,\mu}(\sigma(t_i)) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t_i)$ . And for all  $t_i$  s.t.  $i \notin \mu(f)$ , we have  $\mathcal{I}_{0,\Delta,\mu,f,i}(\sigma(t_i)) = \sigma(t_i)$ . This implies  $f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k))) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .
  - If  $f \notin \Delta$ ,  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = c(f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k))), t')$  for some  $t'$ . Applying a  $\mathcal{C}_\varepsilon$  step to this term, we obtain again the term  $f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k)))$ , and using the previous item result, we get  $f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k))) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .

Then we conclude  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .

The second part of the lemma is proved similarly. If  $t$  is a variable then  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ . Now let  $t = f(t_1, \dots, t_k)$ . Since  $f \in \Delta$ ,  $\mathcal{I}_{0,\Delta,\mu}(\sigma(t)) = \mathcal{I}_{0,\Delta,\mu}(f(\sigma(t_1), \dots, \sigma(t_k))) = f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k)))$ . For  $i \in \mu(f)$ , we have  $\mathcal{I}_{0,\Delta,\mu,f,i}(\sigma(t_i)) = \mathcal{I}_{0,\Delta,\mu}(\sigma(t_i)) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t_i)$  by the induction hypothesis. For  $i \notin \mu(f)$ , we have  $\mathcal{I}_{0,\Delta,\mu,f,i}(\sigma(t_i)) = \sigma(t_i)$ . This implies that  $f(\mathcal{I}_{0,\Delta,\mu,f,1}(\sigma(t_1)), \dots, \mathcal{I}_{0,\Delta,\mu,f,n}(\sigma(t_k))) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},t}(t)$ .

**Proposition 4.** [16] *Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $\Delta \subseteq \mathcal{F}$ . For all  $\mu$ -terminating terms  $t \in \mathcal{T}(\mathcal{F}, \mathcal{X})$ , we have that  $\mathcal{I}_{0,\Delta,\mu}(t) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* t$ .*

*Proof.* By structural induction on  $t$ .

- If  $t = x \in \mathcal{X}$  then  $\mathcal{I}_{0,\Delta,\mu}(x) = x$ .
- If  $t = f(t_1, \dots, t_k)$ , we have that  $\mathcal{I}_{0,\Delta,\mu}(t_i) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* t_i$  if  $i \in \mu(f)$  (by the induction hypothesis), and if  $i \notin \mu(f)$ ,  $t_i$  is maintained inalterable. We have two possibilities:
  - If  $f \in \Delta$  then  $\mathcal{I}_{0,\Delta,\mu}(f(t_1, \dots, t_k)) = f(t'_1, \dots, t'_n)$  where by the induction hypothesis we have that  $t'_i \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* t_i$ . Hence,  $f(t'_1, \dots, t'_n) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* f(t_1, \dots, t_k)$ .
  - If  $f \notin \Delta$  then  $\mathcal{I}_{0,\Delta,\mu}(f(t_1, \dots, t_k)) = c(f(t'_1, \dots, t'_k), t')$  and, applying only  $\mathcal{C}_\varepsilon$  steps we have  $c(f(t'_1, \dots, t'_k), t') \hookrightarrow_{\mathcal{C}_\varepsilon,\mu} f(t'_1, \dots, t'_n) \hookrightarrow_{\mathcal{C}_\varepsilon,\mu}^* f(t_1, \dots, t_k)$ .

**Lemma 4.** *Let  $\tau = (\mathcal{P}, \mathcal{R}, S, \mu)$  be a CS problem where  $\mathcal{P} = (\mathcal{G}, P)$ ,  $\mathcal{R} = (\mathcal{F}, R)$  and  $S = (\mathcal{H}, S)$ . Let  $\mathcal{P} \cup \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, S, \mu)$  be strongly conservative, and  $\Delta = (\mathcal{G} \cup \mathcal{F} \cup \mathcal{H}) \setminus \{\text{root}(\ell) \mid \ell \rightarrow r \in \mathcal{R} \setminus \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, S, \mu)\}$ . If  $s$  and  $t$  are  $\mu$ -terminating on  $\mathcal{R}$  and  $s \hookrightarrow_{\mathcal{R},\mu} t$  then  $\mathcal{I}_{0,\Delta,\mu}(s) \hookrightarrow_{\mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, S, \mu) \cup \mathcal{C}_\varepsilon,\mu}^+ \mathcal{I}_{0,\Delta,\mu}(t)$ .*

*Proof.* By induction on the position  $p$  of the redex in  $s \hookrightarrow_{\{\ell \rightarrow r\},\mu} t$ . First assume that  $\text{root}(s) \in \Delta$  and  $p = \Lambda$  (and therefore  $\ell \rightarrow r \in \mathcal{U}^\blacktriangleright(\mathcal{P}, \mathcal{R}, S, \mu)$ ). So we have  $s = \sigma(\ell) \xrightarrow{\Lambda}_{\{\ell \rightarrow r\},\mu} \sigma(r) = t$  for some substitution  $\sigma$ . Moreover, for all subterms  $r'$  at  $\mu$ -replacing positions of  $r$ ,  $\text{root}(r') \in \Delta$  by definition of  $\Delta$ . Since  $\text{Var}^\mu(\ell) \cap \text{Var}^\#(\ell) = \emptyset$  and  $\text{Var}^\mu(r) \cap \text{Var}^\#(r) = \emptyset$ , we have by Proposition 3 that  $\sigma_{\mathcal{I}_{0,\Delta,\mu}}(\ell) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},\ell}(\ell)$  and  $\sigma_{\mathcal{I}_{0,\Delta,\mu}}(r) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu},r}(r)$ . Moreover, for all

variables  $x$  we have  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,\ell}}(x) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,r}}(x)$ . To see this, note that by strong conservativity,  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,\ell}}$  and  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,r}}$  only differ on variables  $x \in \mathcal{V}ar^\mu(\ell) \setminus \mathcal{V}ar^\mu(r)$ . Here, we have  $\bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,\ell}}(x) = \mathcal{I}_{0,\Delta,\mu}(\sigma(x)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \sigma_{\mathcal{I}_{0,\Delta,\mu}}(x) = \sigma(x) = \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,r}}(x)$  by Lemma 3. Hence,

$$\begin{aligned} \mathcal{I}_{0,\Delta,\mu}(s) &= \mathcal{I}_{0,\Delta,\mu}(\sigma(\ell)) \\ &\hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \sigma_{\mathcal{I}_{0,\Delta,\mu}}(\ell) && \text{by Lemma 3} \\ &= \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,\ell}}(\ell) \\ &\hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,r}}(\ell) \\ &\rightarrow_{\{\ell \rightarrow r\}} \bar{\sigma}_{\mathcal{I}_{0,\Delta,\mu,r}}(r) \\ &= \sigma_{\mathcal{I}_{0,\Delta,\mu}}(r) \\ &= \mathcal{I}_{0,\Delta,\mu}(\sigma(r)) && \text{by Lemma 3} \\ &= \mathcal{I}_{0,\Delta,\mu}(t) \end{aligned}$$

Now let the case where  $\text{root}(s) \in \Delta$  and  $p \neq \Lambda$ . Hence,  $s = f(s_1, \dots, s_i, \dots, s_n)$ ,  $t = f(s_1, \dots, t_i, \dots, s_n)$ ,  $i \in \mu(f)$ , and  $s_i \hookrightarrow_{\{\ell \rightarrow r\}, \mu} t_i$ . The induction hypothesis implies  $\mathcal{I}_{0,\Delta,\mu}(s_i) \hookrightarrow_{\{\ell \rightarrow r\} \cup \mathcal{C}_{\varepsilon,\mu}}^+ \mathcal{I}_{0,\Delta,\mu}(t_i)$  and hence,  $\mathcal{I}_{0,\Delta,\mu}(s) \hookrightarrow_{\{\ell \rightarrow r\} \cup \mathcal{C}_{\varepsilon,\mu}}^+ \mathcal{I}_{0,\Delta,\mu}(t)$ . Finally, we consider the case  $\text{root}(s) \notin \Delta$ . In this case,  $\mathcal{I}_{0,\Delta,\mu}(t) \in \text{order}(\{\mathcal{I}_{0,\Delta,\mu}(u) \mid s \hookrightarrow_{\mathcal{R},\mu} u\})$  because  $s \hookrightarrow_{\mathcal{R},\mu} t$ . By applying  $\mathcal{C}_\varepsilon$  rules, we get  $\mathcal{I}_{0,\Delta,\mu}(s) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^+ \mathcal{I}_{0,\Delta,\mu}(t)$ .

**Definition 20 ( $\mu$ -Interpretation [16]).** Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\Delta \subseteq \mathcal{F}$ . Let  $>$  be an arbitrary total ordering over  $\mathcal{T}(\mathcal{F} \cup \{\perp, c\}, \mathcal{X})$  where  $\perp$  is a fresh constant symbol and  $c$  is a fresh binary symbol (with  $\mu(c) = \{1, 2\}$ ). The  $\mu$ -interpretation  $\mathcal{I}_{1,\Delta,\mu}$  is a mapping from arbitrary terms in  $\mathcal{T}(\mathcal{F}, \mathcal{X})$  to terms in  $\mathcal{T}(\mathcal{F} \cup \{\perp, c\}, \mathcal{X})$  defined as follows:

$$\mathcal{I}_{1,\Delta,\mu}(t) = \begin{cases} t & \text{if } t \in \mathcal{X} \\ f(\mathcal{I}_{1,\Delta,\mu}(t_1), \dots, \mathcal{I}_{1,\Delta,\mu}(t_k)) & \text{if } t = f(t_1, \dots, t_k) \text{ and } f \in \Delta \\ & \text{or } t \text{ is non-}\mu\text{-terminating} \\ c(f(\mathcal{I}_{1,\Delta,\mu}(t_1), \dots, \mathcal{I}_{1,\Delta,\mu}(t_k)), t') & \text{if } t = f(t_1, \dots, t_k) \text{ and } f \notin \Delta \\ & \text{and } t \text{ is } \mu\text{-terminating} \end{cases}$$

where  $t' = \text{order}(\{\mathcal{I}_{1,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\})$

$$\text{order}(T) = \begin{cases} \perp, & \text{if } T = \emptyset \\ c(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } > \end{cases}$$

**Lemma 5.** [16] Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$ ,  $\Gamma = \bigcup_{\ell \rightarrow r} \mathcal{F}un^\mu(r)$ , and  $\Delta \in \mathcal{F}$ . If  $\Gamma \subseteq \Delta$  then  $\mathcal{I}_{1,\Delta,\mu}$  is well-defined.

*Proof.* According to Definition 20, to obtain an infinite term as the result of  $\mathcal{I}_{1,\Delta,\mu}(t)$  for a given term  $t$ , we would have to perform an infinite number of applications of  $\text{order}(\{\mathcal{I}_{1,\Delta,\mu}(u) \mid t \hookrightarrow_{\mathcal{R},\mu} u\})$ . This means that  $t$  is  $\mu$ -terminating and that there exists an infinite sequence (extracted from Definition 20) of the form  $t = u_1 \supseteq t_1 \hookrightarrow_{\mathcal{R},\mu} u_2 \supseteq t_2 \hookrightarrow_{\mathcal{R},\mu} u_3 \dots$  where  $\text{root}(t_i) \notin \Delta$  and  $t_i$  is  $\mu$ -terminating for all  $i \geq 1$ . Without loss of generality we can assume that  $t_i \not\triangleright t_{i+1}$  (otherwise, we simply consider the modified sequence  $\dots \supseteq t_{i-1} \hookrightarrow_{\mathcal{R},\mu} u_i \supseteq t_{i+1} \hookrightarrow_{\mathcal{R},\mu} u_{i+1} \supseteq \dots$ ). For  $i \geq 1$ , there is a rule  $\ell \rightarrow r$ , an active position  $p$

34 Raúl Gutiérrez and Salvador Lucas

of  $t_i$ , and a substitution  $\sigma$  such that  $t_i = C[\sigma(\ell)]_p \hookrightarrow_{\mathcal{R},\mu} C[\sigma(r)]_p = u_{i+1} \triangleright t_{i+1}$ . We have the following possibilities:

- $t_{i+1}$  is a subterm of  $u_{i+1}$  above position  $p$ , i.e.,  $t_{i+1} \triangleright \sigma(r)$ . Then  $u_{i+1} \triangleright_{\mu} t_{i+1}$  because  $p$  is an active position.
- $t_{i+1}$  is a subterm of  $u_{i+1}$  neither above nor below  $p$ . Then we already have  $t_i \triangleright t_{i+1}$  in contradiction to the prerequisite.
- $t_{i+1}$  is a subterm of  $u_{i+1}$  strictly below position  $p$ , i.e.,  $\sigma(r) \triangleright t_{i+1}$ . Note that there is no variable  $x$  in  $r$  such that  $\sigma(x) \triangleright t_{i+1}$ , because that would already imply  $t_i \triangleright t_{i+1}$  in contradiction to the prerequisite. Hence, there is a subterm  $s$  with  $s \notin \mathcal{X}$  and  $r \triangleright s$  such that  $\sigma(s) = t_{i+1}$ . Since  $\text{root}(s) \notin \Delta$  and  $\mathcal{F}un^{\#}(r) \subseteq \Delta$ , we have  $r \triangleright_{\mu} s$  and hence,  $u_{i+1} \triangleright_{\mu} t_{i+1}$ .

The resulting sequence is:  $t = u_1 \triangleright_{\mu} t_1 \hookrightarrow_{\mathcal{R},\mu} u_2 \triangleright_{\mu} t_2 \hookrightarrow_{\mathcal{R},\mu} u_3 \dots$ , contradicting the termination of  $t$ .

**Definition 21.** [16] Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS,  $\mu \in M_{\mathcal{F}}$  and  $\sigma$  be a substitution. Let  $\Delta \subseteq \mathcal{F}$ . We denote by  $\sigma_{\mathcal{I}_{1,\Delta,\mu}}$  a substitution such that, given a term  $t$  replaces occurrences of  $x$  by  $\mathcal{I}_{1,\Delta,\mu}(\sigma(x))$ .

**Lemma 6.** [16] Let  $\mathcal{R} = (\mathcal{F}, R)$  be a TRS and  $\mu \in M_{\mathcal{F}}$ . Let  $\Delta \in \mathcal{F}$ . If all subterms  $t'$  of  $t$  at non- $\mu$ -replacing positions are from  $\mathcal{T}(\Delta, \mathcal{X})$  then we have  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$  and if  $t \in \mathcal{T}(\Delta, \mathcal{X})$  then we have  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ .

*Proof.* We use the induction on  $t$ . If  $t$  is a variable then  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ . Now, let  $t = f(t_1, \dots, t_k)$ . By induction hypothesis  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t_i)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_i)$  for  $1 \leq i \leq n$ . Moreover, by hypothesis: whenever  $i \notin \mu(f)$  then  $t_i$  contains only  $\Delta$ -symbols. Then  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t_i)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_i)$  for all  $i \notin \mu(f)$  by induction hypothesis, and  $f(\mathcal{I}_{1,\Delta,\mu}(\sigma(t_1)), \dots, \mathcal{I}_{1,\Delta,\mu}(\sigma(t_k))) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* f(\sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_1), \dots, \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_k)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ .

- If  $f \in \Delta$  or  $t$  is non- $\mu$ -terminating, then we have that  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) = t'$  where  $t' = f(\mathcal{I}_{1,\Delta,\mu}(\sigma(t_1)), \dots, \mathcal{I}_{1,\Delta,\mu}(\sigma(t_k)))$  then we  $\mu$ -rewrite  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* f(\sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_1), \dots, \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_k)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ . If there are only  $\Delta$ -symbols in  $t$ , then, by the induction hypothesis,  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t_i)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t_i)$  for all  $i$ ,  $1 \leq i \leq n$ , hence  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) = \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ .
- If  $f \notin \Delta$  and  $t$  is  $\mu$ -terminating, then  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) = c(t', t'') \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* t'$  for some term  $t''$ , with  $t' = f(\mathcal{I}_{1,\Delta,\mu}(\sigma(t_1)), \dots, \mathcal{I}_{1,\Delta,\mu}(\sigma(t_k)))$ . Hence,  $\mathcal{I}_{1,\Delta,\mu}(\sigma(t)) \hookrightarrow_{\mathcal{C}_{\varepsilon,\mu}}^* \sigma_{\mathcal{I}_{1,\Delta,\mu}}(t)$ .

**Lemma 7.** Let  $\tau = (\mathcal{P}, \mathcal{R}, S, \mu)$  be a CS problem where  $\mathcal{P} = (\mathcal{G}, P)$ ,  $\mathcal{R} = (\mathcal{F}, R)$  and  $S = (\mathcal{H}, S)$ . Let  $\Delta = (\mathcal{G} \cup \mathcal{F} \cup \mathcal{H}) \setminus \{\text{root}(\ell) \mid \ell \rightarrow r \in \mathcal{R} \setminus \mathcal{U}^{\mathcal{P}}(\mathcal{P}, \mathcal{R}, S, \mu)\}$ . If  $s$  and  $t$  are  $\mu$ -terminating on  $\mathcal{R}$  and  $s \hookrightarrow_{\mathcal{R},\mu} t$  then  $\mathcal{I}_{1,\Delta,\mu}(s) \hookrightarrow_{\mathcal{U}^{\mathcal{P}}(\mathcal{P}, \mathcal{R}, S, \mu) \cup \mathcal{C}_{\varepsilon,\mu}}^+ \mathcal{I}_{1,\Delta,\mu}(t)$ .

*Proof.* We proceed by induction on the position  $p \in \text{Pos}^{\mu}(s)$  of the redex in the reduction  $s \hookrightarrow_{\{\ell \rightarrow r\},\mu} t$ . First let  $\text{root}(s) \in \Delta$  and  $p = \Lambda$  ( $\ell \rightarrow r \in$

$\mathcal{U}^\circ(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ . So we have  $s = \sigma(\ell) \xrightarrow{\Delta}_{\{\ell \rightarrow r\}, \mu} \sigma(r) = t$  for some substitution  $\sigma$ . Moreover,  $r \in \mathcal{T}(\Delta, \mathcal{X})$  and all subterms  $\ell'$  of  $\ell$  at inactive positions are from  $\mathcal{T}(\Delta, \mathcal{X})$  by the conditions on  $\Delta$ . By Lemma 6 we get  $\mathcal{I}_{1, \Delta, \mu}(\sigma(\ell)) \xrightarrow{\mathcal{C}_\varepsilon, \mu} \sigma \mathcal{I}_{1, \Delta, \mu}(\ell) \xrightarrow{\{\ell \rightarrow r\}, \mu} \sigma \mathcal{I}_{1, \Delta, \mu}(r) = \mathcal{I}_{1, \Delta, \mu}(\sigma(r))$ . Now let  $\text{root}(s) \in \Delta$  and  $p \neq \Delta$ . Hence,  $s = f(s_1, \dots, s_i, \dots, s_n)$ ,  $t = f(s_1, \dots, t_i, \dots, s_n)$ ,  $i \in \mu(f)$ , and  $s_i \xrightarrow{\{\ell \rightarrow r\}, \mu} t_i$ . The induction hypothesis implies  $\mathcal{I}_{1, \Delta, \mu}(s_i) \xrightarrow{\{\ell \rightarrow r\} \cup \mathcal{C}_\varepsilon, \mu} \mathcal{I}_{1, \Delta, \mu}(t_i)$  and hence,  $\mathcal{I}_{1, \Delta, \mu}(s) \xrightarrow{\{\ell \rightarrow r\} \cup \mathcal{C}_\varepsilon, \mu} \mathcal{I}_{1, \Delta, \mu}(t)$ . Finally, we consider the case  $\text{root}(s) \notin \Delta$ . In this case,  $\mathcal{I}_{1, \Delta, \mu}(t) \in \text{order}(\{\mathcal{I}_{1, \Delta, \mu}(u) \mid s \xrightarrow{\mathcal{R}, \mu} u\})$  because  $s \xrightarrow{\mathcal{R}, \mu} t$ . By applying  $\mathcal{C}_\varepsilon$  rules, we get  $\mathcal{I}_{1, \Delta, \mu}(s) \xrightarrow{\mathcal{C}_\varepsilon, \mu} \mathcal{I}_{1, \Delta, \mu}(t)$ .

**Theorem 10 ( $\mu$ -Reduction Triple Processor with Usable Rules).** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem where  $\mathcal{P} = (\mathcal{G}, P)$ ,  $\mathcal{R} = (\mathcal{F}, R)$  and  $\mathcal{S} = (\mathcal{H}, S)$ . Let  $(\succ, \sqsupset, \succeq)$  be a  $\mu$ -reduction triple such that*

1.  $\mathcal{P} \subseteq \succ \cup \sqsupset$ ,
2. at least one of the following holds:
  - (a)  $\mathcal{U}^\bullet(\tau) \subseteq \succ$ ,  $\mathcal{P} \cup \mathcal{U}^\bullet(\tau)$  is strongly  $\mu$ -conservative,  $\succ$  is  $\mathcal{C}_\varepsilon$ -compatible
  - (b)  $\mathcal{U}^\circ(\tau) \subseteq \succ$ ,  $\succ$  is  $\mathcal{C}_\varepsilon$ -compatible,
  - (c)  $\mathcal{R} \subseteq \succ$ ,
3. and, whenever  $\mathcal{P}_\mathcal{X} \neq \emptyset$  we have that  $\mathcal{S} \subseteq \succ \cup \sqsupset \cup \succeq$ .

Let  $\mathcal{P}_\sqsupset = \{u \rightarrow v \in \mathcal{P} \mid u \sqsupset v\}$  and  $\mathcal{S}_\sqsupset = \{s \rightarrow t \in \mathcal{S} \mid s \sqsupset t\}$ . Then, the processor  $\text{Proc}_{UR}$  given by

$$\text{Proc}_{UR}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_\sqsupset, \mu)\} & \text{if (1), (2) and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

*Proof.* Completeness is obvious, since  $\mathcal{P} \setminus \mathcal{P}_\sqsupset \subseteq \mathcal{P}$  and  $\mathcal{S} \setminus \mathcal{S}_\sqsupset \subseteq \mathcal{S}$ . Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$ , but there is no infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_\sqsupset, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_\sqsupset, \mu)$ -chain. Due to the finiteness of  $\mathcal{P}$  and  $\mathcal{S}$ , we can assume that there are subsets  $\mathcal{Q} \subseteq \mathcal{P}$  and  $\mathcal{T} \subseteq \mathcal{S}$  such that  $A$  has a tail  $B$

$$\sigma(u_1) \left\{ \xrightarrow{\mathcal{Q}_\mathcal{X}, \mu} \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \right\} t'_1 \xrightarrow{\mathcal{R}, \mu} \sigma(u_2) \left\{ \xrightarrow{\mathcal{Q}_\mathcal{X}, \mu} \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \right\} \dots$$

for some substitution  $\sigma$ , where all pairs in  $\mathcal{Q}$  and all rules in  $\mathcal{T}$  are infinitely often used (note that, if  $\mathcal{T} \neq \emptyset$ , then  $\mathcal{T}_\triangleright \neq \emptyset$  and  $\mathcal{Q}_\mathcal{X} \neq \emptyset$ ), and, for all  $i \geq 1$ , (1) if  $u_i \rightarrow v_i \in \mathcal{Q}_\mathcal{G}$ , then  $t'_i = \sigma(v_i)$  and (2) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_\mathcal{X}$ , then  $\sigma(u_i) \xrightarrow{\mathcal{Q}_\mathcal{X}} \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_\triangleright, \mu} t'_i$ . Moreover, all  $t'_i$  are  $\mu$ -terminating.

First, we consider Item 2a. If  $\mathcal{P} \cup \mathcal{U}^\bullet(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  is strongly conservative, we have no collapsing pairs, that is, we can write:

$$\sigma(u_1) \xrightarrow{\mathcal{Q}_\mathcal{G}, \mu} t'_1 \xrightarrow{\mathcal{R}, \mu} \sigma(u_2) \xrightarrow{\mathcal{Q}_\mathcal{G}, \mu} \dots$$



2. If  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_X$  (which is not empty whenever  $\mathcal{T}$  is not empty), then  $\sigma_{\mathcal{I}_1, \Delta, \mu}(v_i) = \sigma_{\mathcal{I}_1, \Delta, \mu}(x_i) \xrightarrow{\Delta}_{\mathcal{T}_{\triangleright \mu}, \mu}^* \sigma_{\mathcal{I}_1, \Delta, \mu}(\ell_i)$ . Since  $\ell_j (\succeq \cup \succ \cup \sqsupset) r_j$  for all  $\ell_j \rightarrow r_j \in \mathcal{T}_{\triangleright \mu}$ , we have  $\sigma_{\mathcal{I}_1, \Delta, \mu}(v_i) = \sigma_{\mathcal{I}_1, \Delta, \mu}(x_i) (\succeq \cup \succ \cup \sqsupset) \sigma_{\mathcal{I}_1, \Delta, \mu}(\ell_i)$ . Furthermore, we are assuming that  $\ell (\succeq \cup \succ \cup \sqsupset) r$  for all  $\ell \rightarrow r \in \mathcal{T}_{\sharp}$ . Since  $\sigma_{\mathcal{I}_1, \Delta, \mu}(\ell_i) \xrightarrow{\Delta}_{\mathcal{T}_{\sharp}} \sigma_{\mathcal{I}_1, \Delta, \mu}(r_i)$ , for some  $\ell_i \rightarrow r_i \in \mathcal{T}_{\sharp}$  and, by stability of the (quasi-)orderings we have that  $\sigma_{\mathcal{I}_1, \Delta, \mu}(\ell_i) (\succeq \cup \succ \cup \sqsupset) \sigma_{\mathcal{I}_1, \Delta, \mu}(r_i)$ . Hence, by transitivity of  $\succ$  (and compatibility of  $\succ$ ,  $\sqsupset$  and  $\succeq$ ), we have  $\sigma_{\mathcal{I}_1, \Delta, \mu}(v_i) = \sigma_{\mathcal{I}_1, \Delta, \mu}(x_i) (\succeq \cup \succ \cup \sqsupset) \sigma_{\mathcal{I}_1, \Delta, \mu}(r_i)$ . Since  $\sigma_{\mathcal{I}_1, \Delta, \mu}(r_i) \xrightarrow{\Delta}_{\mathcal{U}^{\circ}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) \cup \mathcal{C}_{\varepsilon, \mu}} \sigma_{\mathcal{I}_1, \Delta, \mu}(u_{i+1})$ , we also have that, for all  $i \geq 1$ ,  $\sigma_{\mathcal{I}_1, \Delta, \mu}(r_i) \succ \sigma(u_{i+1})$ . Therefore, again by transitivity of  $\succ$  and compatibility of  $\succ$ ,  $\succeq$  and  $\sqsupset$ , we conclude that  $\sigma_{\mathcal{I}_1, \Delta, \mu}(u_i) (\succeq \cup \succ \cup \sqsupset) \sigma_{\mathcal{I}_1, \Delta, \mu}(r_i) \succ \sigma_{\mathcal{I}_1, \Delta, \mu}(u_{i+1})$  and hence  $\sigma_{\mathcal{I}_1, \Delta, \mu}(u_i) (\succ \cup \sqsupset) \sigma_{\mathcal{I}_1, \Delta, \mu}(u_{i+1})$ .

Since  $u \rightarrow v$  and  $\ell \rightarrow r$  occur infinitely often in  $B$ , there is an infinite set  $\mathcal{I} \subseteq \mathbb{N}$  of pairs such that  $\sigma_{\mathcal{I}_1, \Delta, \mu}(u_i) \sqsupset \sigma_{\mathcal{I}_1, \Delta, \mu}(u_{i+1})$  for all  $i \in \mathcal{I}$ . Thus, by using the compatibility conditions of the  $\mu$ -reduction triple, we obtain an infinite decreasing  $\sqsupset$ -sequence which contradicts well-foundedness of  $\sqsupset$ .

Therefore,  $B$  is an infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\sqsupset}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\sqsupset}, \mu)$ -chain, thus leading to a contradiction.

Finally, Item 2c is proved in Theorem 9.

## J Proof of Theorem 11

**Theorem 11 (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)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_X \neq \emptyset$ , then for all  $s \rightarrow t \in \mathcal{S}_{\sharp}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\sharp}\}$ . Let  $\mathcal{P}_{\pi, \triangleright \mu} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) \triangleright_{\mu} \pi(v)\}$  and  $\mathcal{S}_{\pi, \triangleright \mu} = \mathcal{S}_{\triangleright \mu} \cup \{s \rightarrow t \in \mathcal{S}_{\sharp} \mid s \triangleright_{\mu} \pi(t)\}$ . Then,  $\text{Proc}_{\text{subterm}}$  given by*

$$\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright \mu}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright \mu}, \mu)\} & \text{if } \pi(\mathcal{P}) \subseteq \triangleright_{\mu} \\ & \text{and whenever } \mathcal{P}_X \neq \emptyset, \\ & \text{then } \mathcal{S}_{\pi} \subseteq \triangleright_{\mu} \\ \{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

*Proof.* Completeness is obvious because  $\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright \mu} \subseteq \mathcal{P}$  and  $\mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright \mu} \subseteq \mathcal{S}$ . For soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$  but there is no infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright \mu}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright \mu}, \mu)$ -chain. Since  $\mathcal{P}$  and  $\mathcal{S}$  are finite, we can assume that there are subsets  $\mathcal{Q} \subseteq \mathcal{P}$  and  $\mathcal{T} \subseteq \mathcal{S}$  such that  $A$  has a tail  $B$  which is an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mathcal{T}, \mu)$ -chain where all pairs in  $\mathcal{Q}$  and rules in  $\mathcal{T}$  are infinitely often used. Note that, if  $\mathcal{T} \neq \emptyset$ , then  $\mathcal{T}_{\sharp} \neq \emptyset$  and  $\mathcal{Q}_X \neq \emptyset$ . Assume that  $B$  is as follows:

$$\sigma(u_1) \left\{ \begin{array}{c} \xrightarrow{\Delta}_{\mathcal{Q}_{\mathcal{G}}, \mu} \\ \xrightarrow{\Delta}_{\mathcal{I}_{\triangleright \mu}, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_{\sharp}, \mu} \end{array} \right\} t_1 \xrightarrow{\Delta}_{\mathcal{R}, \mu}^* \sigma(u_2) \left\{ \begin{array}{c} \xrightarrow{\Delta}_{\mathcal{Q}_{\mathcal{G}}, \mu} \\ \xrightarrow{\Delta}_{\mathcal{I}_{\triangleright \mu}, \mu} \circ \xrightarrow{\Delta}_{\mathcal{T}_{\sharp}, \mu} \end{array} \right\} \dots$$

38 Raúl Gutiérrez and Salvador Lucas

for some substitution  $\sigma$ , and, for all  $i \geq 1$ , (A) if  $u_i \rightarrow v_i \in \mathcal{Q}_G$ , then  $t_i = \sigma(v_i)$  and (B) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_X$  and  $\ell_i \rightarrow r_i \in \mathcal{T}_\sharp$ , then  $\sigma(u_i) \xrightarrow{\mathcal{Q}_X} \sigma(x_i) \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} \sigma(\ell_i) \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} \sigma(r_i) = t_i$ . Note that, for all  $i \geq 1$ ,

- $\text{root}(\sigma(u_i)) \in \text{Root}(\mathcal{P})$  because  $\text{root}(u_i) \in \text{Root}(\mathcal{P})$ ,
- If  $u_i \rightarrow v_i \in \mathcal{P}_G$ , then  $t_i = \sigma(v_i)$ . By (2),  $v_i \notin \mathcal{X}$ . Therefore,  $\text{root}(v_i) \in \text{Root}(\mathcal{P})$ .
- If  $u_i \rightarrow v_i \in \mathcal{P}_X$ , then  $t_i = \sigma(r_i)$ . By (3),  $\text{root}(r_i) \in \text{Root}(\mathcal{P})$ . Therefore,  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ .

Since  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$  for all  $i \geq 0$  and  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$  (due to (1)), we can actually write  $t_i \xrightarrow{\mathcal{R}, \mu}^* \sigma(u_{i+1})$  because  $\mu$ -rewritings with  $\mathcal{R}$  cannot change  $\text{root}(t_i)$ . Hence  $\pi(t_i) \xrightarrow{\mathcal{R}, \mu}^* \pi(\sigma(u_{i+1}))$  and also  $\text{root}(t_i) = \text{root}(\sigma(u_{i+1}))$  for all  $i \geq 1$ . Since  $\pi(u_i) \triangleright_\mu \pi(v_i)$  for all  $i \geq 1$ , by stability of  $\triangleright_\mu$ , we have, for all  $i \geq 1$

- If  $u_i \rightarrow v_i \in \mathcal{Q}_G$ , then  $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \triangleright_\mu \sigma(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$ .
- If  $u_i \rightarrow x_i \in \mathcal{Q}_X$ , then  $\sigma(x_i) \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} t'_i \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} t_i$ . We prove by induction on the length  $m$  of the sequence  $\sigma(x_i) \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} t'_i$  that  $\sigma(x_i) \triangleright_\mu t'_i$ :
  - If  $m = 0$  then  $\sigma(x_i) = t'_i$ .
  - If  $m > 1$ , then we can write  $\sigma(x_i) \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} t''_i \xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*} t'_i$  where  $t''_i$  is

obtained from  $\sigma(x_i)$  in  $m-1$   $\xrightarrow{\mathcal{T}_{\triangleright_\mu, \mu}^*}$ -steps. By the induction hypothesis,  $\sigma(x_i) \triangleright_\mu t''_i = \sigma(\ell)$  for some  $\ell \rightarrow r \in \mathcal{T}_{\triangleright_\mu} \subseteq \mathcal{S}_{\triangleright_\mu}$ . Since  $\ell \triangleright_\mu r$  (by definition of  $\mathcal{S}_{\triangleright_\mu}$ ), then by stability of  $\triangleright_\mu$  we have  $\sigma(\ell) \triangleright_\mu \sigma(r) = t'_i$  and, hence,  $\sigma(x_i) \triangleright_\mu t'_i$ .

Since  $t'_i = \sigma(\ell_i)$ , we have that  $t_i = \sigma(r_i)$ . Since  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ , and  $\ell_i \triangleright_\mu \pi(r_i)$  (by the assumption  $\mathcal{S}_\pi \subseteq \triangleright_\mu$ ), by stability of  $\triangleright_\mu$ , we have  $t'_i = \sigma(\ell_i) \triangleright_\mu \sigma(\pi(r_i)) = \pi(\sigma(r_i)) = \pi(t_i)$  and, hence,

$$\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \triangleright_\mu \sigma(\pi(x_i)) = \sigma(x_i) \triangleright_\mu \sigma(\ell_i) \triangleright_\mu \sigma(\pi(r_i)) = \pi(t_i)$$

Therefore, we have  $\pi(\sigma(u_i)) \triangleright_\mu \pi(t_i)$  in any case. No pair  $u \rightarrow v \in \mathcal{Q}$  satisfies that  $\pi(u) \triangleright_\mu \pi(v)$  and no rule  $\ell \rightarrow r \in \mathcal{T}$  satisfies  $\pi(\ell) \triangleright_\mu \pi(r)$ . Otherwise, we get a contradiction in both of the following two complementary cases:

1. if  $\pi(f) \notin \mu(f)$  for all  $f \in \text{Root}(\mathcal{Q})$ , then, for all  $i \geq 0$ ,  $\pi(t_i) = \pi(\sigma(u_{i+1}))$ , because no  $\mu$ -rewritings are possible on the  $\pi(\text{root}(t_i))$ -th immediate subterm  $\pi(t_i)$  of  $t_i$ . Since  $\pi(\sigma(u_{i+1})) \triangleright_\mu \pi(t_{i+1})$ , we have that  $\pi(t_i) \triangleright_\mu \pi(t_{i+1})$  for all  $i \geq 0$ . Furthermore, since we assume  $\pi(u) \triangleright_\mu \pi(v)$  for some  $u \rightarrow v \in \mathcal{Q}$ ,  $\ell \triangleright_\mu r$  for some  $\ell \rightarrow r \in \mathcal{T}_{\triangleright_\mu}$ , or  $\ell \triangleright_\mu \pi(r)$  for some  $\ell \rightarrow r \in \mathcal{T}_\sharp$  which apply infinitely often in  $B$  (remind that  $\mathcal{T} = \emptyset$  if and only if  $\mathcal{Q}_X = \emptyset$ ), and by stability of  $\triangleright_\mu$ , there is a maximal infinite set  $J = \{j_1, j_2, \dots\} \subseteq \mathbb{N}$  such that  $\pi(t_{j_i}) \triangleright_\mu \pi(t_{j_{i+1}})$  for all  $i \geq 1$ . Thus, we obtain an infinite sequence  $\pi(t_{j_1}) \triangleright_\mu \pi(t_{j_2}) \triangleright_\mu \dots$  which contradicts the well-foundedness of  $\triangleright_\mu$ .

2. if  $\pi(f) \in \mu(f)$  for some  $f \in \text{Root}(\mathcal{Q})$ , then, since  $\text{root}(t_i) = \text{root}(\sigma(u_{i+1}))$  and all pairs in  $\mathcal{Q}$  occur infinitely often in  $B$ , we can assume that  $\text{root}(t_1) = f$ . Furthermore, since  $A$  is minimal, we can assume that  $t_1$  is  $\mu$ -terminating (w.r.t.  $\mathcal{R}$ ). Since  $\pi(t_i) \xrightarrow{\ast}_{\mathcal{R}, \mu} \pi(\sigma(u_{i+1}))$  and  $\pi(\sigma(u_{i+1})) \triangleright_{\mu} \pi(t_{i+1})$  for all  $i \geq 0$ , the sequence  $B$  is transformed into an infinite  $\xrightarrow{\ast}_{\mathcal{R}, \mu} \cup \triangleright_{\mu}$ -sequence

$$\pi(t_1) \xrightarrow{\ast}_{\mathcal{R}, \mu} \pi(\sigma(u_2)) \triangleright_{\mu} \pi(t_2) \xrightarrow{\ast}_{\mathcal{R}, \mu} \pi(\sigma(u_3)) \triangleright_{\mu} \pi(t_3) \xrightarrow{\ast}_{\mathcal{R}, \mu} \dots$$

containing infinitely many  $\triangleright_{\mu}$ -steps, due to  $\pi(u) \triangleright_{\mu} \pi(v)$  for some  $u \rightarrow v \in \mathcal{Q}$ ,  $\ell \triangleright_{\mu} r$  for some  $\ell \rightarrow r \in \mathcal{T}_{\triangleright_{\mu}}$ , or  $\ell \triangleright_{\mu} \pi(r)$  for some  $\ell \rightarrow r \in \mathcal{T}_{\sharp}$  which apply infinitely often in  $B$ . Since  $\triangleright_{\mu}$  is well-founded, the infinite sequence must also contain infinitely many  $\xrightarrow{\ast}_{\mathcal{R}, \mu}$ -steps. By making repeated use of the fact that  $\triangleright_{\mu} \circ \xrightarrow{\ast}_{\mathcal{R}, \mu} \subseteq \xrightarrow{\ast}_{\mathcal{R}, \mu} \circ \triangleright_{\mu}$ , we obtain an infinite  $\xrightarrow{\ast}_{\mathcal{R}, \mu}$ -sequence starting from  $\pi(t_1)$ . Thus,  $\pi(t_1)$  is not  $\mu$ -terminating with respect to  $\mathcal{R}$ . Since  $\pi(f) \in \mu(f)$  and hence  $t_1 \triangleright_{\mu} \pi(t_1)$ , this implies that  $t_1$  is not  $\mu$ -terminating. This contradicts  $\mu$ -termination of  $t_1$ .

Therefore,  $\mathcal{Q} \subseteq \mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright_{\mu}}$  and  $\mathcal{T} \subseteq \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright_{\mu}}$ . Hence,  $B$  is an infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\pi, \triangleright_{\mu}}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, \triangleright_{\mu}}, \mu)$ -chain. This contradicts our initial argument.

## K Proof of Theorem 12

**Theorem 12 (Non- $\mu$ -Replacing Projection Processor).** *Let  $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$  be a CS problem where  $\mathcal{R} = (\mathcal{C} \uplus \mathcal{D}, R)$  and  $\mathcal{P} = (\mathcal{G}, P)$ . Assume that (1)  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$ , (2) the rules in  $\mathcal{P}_{\mathcal{G}}$  are noncollapsing, and (3) if  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , then for all  $s \rightarrow t \in \mathcal{S}_{\sharp}$ ,  $\text{root}(t) \in \text{Root}(\mathcal{P})$ . Let  $\pi$  be a simple projection for  $\mathcal{P}$ . Let  $\mathcal{S}_{\pi} = \mathcal{S}_{\triangleright_{\mu}} \cup \{s \rightarrow \pi(t) \mid s \rightarrow t \in \mathcal{S}_{\sharp}\}$ . Let  $\succsim$  be a stable quasi-ordering on terms whose strict and stable part  $>$  is well-founded such that*

1. for all  $f \in \text{Root}(\mathcal{P})$ ,  $\pi(f) \notin \mu(f)$ ,
2.  $\pi(\mathcal{P}) \subseteq \succsim$ , and,
3. whenever  $\mathcal{S} \neq \emptyset$  and  $\mathcal{P}_{\mathcal{X}} \neq \emptyset$ , we have that  $\mathcal{S}_{\pi} \subseteq \succsim$

Let  $\mathcal{P}_{\pi, >} = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$  and  $\mathcal{S}_{\pi, >} = \{s \rightarrow t \in \mathcal{S}_{\triangleright_{\mu}} \mid s > t\} \cup \{s \rightarrow t \in \mathcal{S}_{\sharp} \mid s > \pi(t)\}$ . Then, the processor  $\text{Proc}_{NRP}$  given by

$$\text{Proc}_{NRP}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) = \begin{cases} \{(\mathcal{P} \setminus \mathcal{P}_{\pi, >}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, >}, \mu)\} & \text{if (1), (2), and (3) hold} \\ \{(\mathcal{P}, \mathcal{R}, \mu)\} & \text{otherwise} \end{cases}$$

is sound and complete.

*Proof.* Completeness is obvious because  $\mathcal{P} \setminus \mathcal{P}_{\pi, >} \subseteq \mathcal{P}$  and  $\mathcal{S} \setminus \mathcal{S}_{\pi, >} \subseteq \mathcal{S}$ . For soundness, we proceed by contradiction. Assume that there is an infinite minimal  $(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ -chain  $A$  but there is no infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\pi, >}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, >}, \mu)$ -chain. Since  $\mathcal{P}$  and  $\mathcal{S}$  are finite, we can assume that there are subsets  $\mathcal{Q} \subseteq \mathcal{P}$  and  $\mathcal{T} \subseteq \mathcal{S}$  such that  $A$  has a tail  $B$  which is an infinite minimal  $(\mathcal{Q}, \mathcal{R}, \mathcal{T}, \mu)$ -chain

40 Raúl Gutiérrez and Salvador Lucas

where all pairs in  $\mathcal{Q}$  and rules in  $\mathcal{T}$  are infinitely often used. Note that, if  $\mathcal{T} \neq \emptyset$ , then  $\mathcal{T}_{\sharp} \neq \emptyset$  and  $\mathcal{Q}_{\mathcal{X}} \neq \emptyset$ . Assume that  $B$  is as follows:

$$\sigma(u_1) \left\{ \begin{array}{c} \xrightarrow{\mathcal{Q}_{\mathcal{G},\mu}} \\ \xrightarrow{\mathcal{Q}_{\mathcal{X},\mu} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{G},\mu}} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} \end{array} \right\} t_1 \xrightarrow{\mathcal{R},\mu}^* \sigma(u_2) \left\{ \begin{array}{c} \xrightarrow{\mathcal{Q}_{\mathcal{G},\mu}} \\ \xrightarrow{\mathcal{Q}_{\mathcal{X},\mu} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{G},\mu}} \circ \overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} \end{array} \right\} \cdots$$

for some substitution  $\sigma$ , and, for all  $i \geq 1$ , (A) if  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$  and (B) if  $u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$  and  $\ell_i \rightarrow r_i \in \mathcal{T}_{\sharp}$ , then  $\sigma(u_i) \xrightarrow{\mathcal{Q}_{\mathcal{X}}} \sigma(x_i) \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}}^* v_i \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} \sigma(\ell_i) \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} \sigma(r_i) = t_i$ . Note that, for all  $i \geq 1$ ,

- $\text{root}(\sigma(u_i)) \in \text{Root}(\mathcal{P})$  because  $\text{root}(u_i) \in \text{Root}(\mathcal{P})$ .
- If  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{G}}$ , then  $t_i = \sigma(v_i)$ . By (2),  $v_i \notin \mathcal{X}$ . Therefore,  $\text{root}(v_i) \in \text{Root}(\mathcal{P})$ .
- If  $u_i \rightarrow v_i \in \mathcal{P}_{\mathcal{X}}$ , then  $t_i = \sigma(r_i)$ . By (3),  $\text{root}(r_i) \in \text{Root}(\mathcal{P})$ . Therefore,  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ .

Since  $t_i \xrightarrow{\mathcal{R},\mu}^* \sigma(u_{i+1})$  for all  $i \geq 0$  and  $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$  (due to (1)), we can actually write  $t_i \xrightarrow{\mathcal{R},\mu}^* \sigma(u_{i+1})$  because  $\mu$ -rewritings with  $\mathcal{R}$  cannot change  $\text{root}(t_i)$  and  $\text{root}(t_i) = \text{root}(\sigma(u_{i+1}))$ . Since  $\pi(u_i) \succsim \pi(v_i)$  for all  $i \geq 1$ , by stability of  $\succsim$ , we have, for all  $i \geq 1$

- If  $u_i \rightarrow v_i \in \mathcal{Q}_{\mathcal{G}}$ , then  $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \succsim \sigma(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$ .
- If  $u_i \rightarrow x_i \in \mathcal{Q}_{\mathcal{X}}$ , then  $\sigma(x_i) \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}}^* t'_i \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} t_i$ . We prove by induction on the length  $m$  of the sequence  $\sigma(x_i) \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}}^* t'_i$  that  $\sigma(x_i) \succsim t'_i$ :
  - If  $m = 0$  then  $\sigma(x_i) = t'_i$ .
  - If  $m > 1$ , then we can write  $\sigma(x_i) \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}}^* t''_i \xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}} t'_i$  where  $t''_i$  is obtained from  $\sigma(x_i)$  in  $m-1$   $\xrightarrow{\overset{\Delta}{\mathcal{T}_{\mathcal{D},\mu,\mu}}}$ -steps. By the induction hypothesis,  $\sigma(x_i) \succsim t''_i = \sigma(\ell)$  for some  $\ell \rightarrow r \in \mathcal{T}_{\mathcal{D},\mu} \subseteq \mathcal{S}_{\mathcal{D},\mu}$ . Since  $\ell \succsim r$  (because  $\mathcal{S}_{\mathcal{D},\mu} \subseteq \mathcal{S}_{\pi} \subseteq \succsim$ ), then by stability of  $\succsim$  we have  $\sigma(\ell) \succsim \sigma(r) = t'_i$  and, hence,  $\sigma(x_i) \succsim t'_i$ .

Since  $t'_i = \sigma(\ell_i)$ , we have that  $t_i = \sigma(r_i)$ . Since  $\text{root}(t_i) \in \text{Root}(\mathcal{P})$ , and  $\ell_i \succsim \pi(r_i)$  (by the assumption  $\mathcal{S}_{\pi} \subseteq \succsim$ ), by stability of  $\succsim$ , we have  $t'_i = \sigma(\ell_i) \succsim \sigma(\pi(r_i)) = \pi(\sigma(r_i)) = \pi(t_i)$  and, hence,

$$\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \succsim \sigma(\pi(x_i)) = \sigma(x_i) \succsim \sigma(\ell_i) \succsim \sigma(\pi(r_i)) = \pi(t_i).$$

Therefore, we have  $\pi(\sigma(u_i)) \succsim \pi(t_i)$  in any case. No pair  $u \rightarrow v \in \mathcal{Q}$  satisfies that  $\pi(u) > \pi(v)$ , no rule  $\ell \rightarrow r \in \mathcal{T}_{\mathcal{D},\mu}$  satisfies that  $\ell > r$ , and no rule  $\ell \rightarrow r \in \mathcal{T}_{\sharp}$  satisfies that  $\ell > \pi(r)$ . Otherwise, by applying the simple projection  $\pi$  to the sequence  $B$ , we get a contradiction as follows:

1. Since  $\pi(f) \notin \mu(f)$  for all  $f \in \text{Root}(\mathcal{Q})$ , no  $\mu$ -rewritings are possible on the subterm  $\pi(t_i)$  of  $t_i$ . Therefore, for all  $i \geq 1$ ,  $\pi(t_i) = \pi(\sigma(u_{i+1})) = \sigma(\pi(u_{i+1}))$ .
2. Due to  $\pi(u_i) \succsim \pi(v_i)$  and by stability of  $\succsim$ , we have that  $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \succsim \sigma(\pi(v_i)) = \pi(\sigma(v_i))$ . Now, we distinguish two cases:

Proving Termination in the CSDP Framework 41

- (a) If  $u_i \rightarrow v_i \in \mathcal{Q}_G$ , then  $\pi(t_i) = \pi(\sigma(v_i)) = \sigma(\pi(v_i))$ . Thus,  $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$ .
- (b) If  $u_i \rightarrow v_i \in \mathcal{Q}_X$ , then  $\sigma(\pi(v_i)) = \sigma(x_i)$ . Thus,  $\sigma(x_i) \gtrsim \sigma(\ell_i)$  and  $\sigma(\ell_i) \gtrsim \sigma(\pi(r_i)) = \pi(\sigma(r_i)) = \pi(t_i)$  (by (3)). Thus,  $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$ .

Thus, we always have  $\pi(\sigma(u_i)) \gtrsim \pi(t_i)$ . We obtain an infinite  $\gtrsim$  sequence

$$\pi(\sigma(u_1)) \gtrsim \pi(t_1) = \pi(\sigma(u_2)) \gtrsim \pi(t_2) \cdots$$

Since pairs in  $\mathcal{Q}$  and rules in  $\mathcal{T}$  occur infinitely often, this sequence contains infinitely many  $>$  steps starting from  $\pi(\sigma(u_1))$ . This contradicts the well-foundedness of  $>$ .

Therefore,  $\mathcal{Q} \subseteq \mathcal{P} \setminus \mathcal{P}_{\pi, >}$  and  $\mathcal{T} \subseteq \mathcal{S} \setminus \mathcal{S}_{\pi, >}$ , i.e.,  $B$  is an infinite minimal  $(\mathcal{P} \setminus \mathcal{P}_{\pi, >}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\pi, >}, \mu)$ -chain. This contradicts our initial assumption.