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.
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 César, 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.
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 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 términos. 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.
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 reescritura. Avui dia, el **marc de parells de dependència** (que desenvolupa aquesta idea) és l’aproximació més reeixida per a provar la terminació de la reescritura. 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 reescritura.

La **reescritura sensible al context** [Luc98, Luc02] és una restricció de la reescritura 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 reescritura sensible al context és útil per a analitzar i provar la terminació de programes en diversos llenguatges de programació i variants de sistemes de reescritura de termes. En els últims quinze anys s’han desenvolupat i programat moltes tècniques per a provar la terminació de la reescritura 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 reescritura 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 reescritura sensible al context.
Contents

1 Introduction 1
   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

I Summary of the Research 19

2 Preliminaries 21
   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

3 Structure of Infinite μ-Rewrite Sequences 27
   3.1 Minimal Non-μ-Terminating Terms 28
   3.2 Infinite μ-Rewrite Sequences Starting from Minimal Terms 29
   3.3 Hidden Terms 31
   3.4 Infinite μ-Rewrite Sequences Starting from Strongly Minimal Terms 35
   3.5 Historical Development of Infinite μ-Rewrite Sequences 36
## Contents

4  **Context-Sensitive Dependency Pairs**  
   4.1 Defining CSDPs .......................... 40  
   4.2 Unhiding TRS .......................... 42  
   4.3 Chains of CSDPs ........................ 45  
   4.4 Historical Development of CSDPs ........... 46

5  **Context-Sensitive Dependency Pair Framework**  
   5.1 Historical Development of the CSDP Framework .... 53

6  **CS Processors**  
   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

7  **Implementing the CSDP Framework**  
   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

8  **Experiments**  
   8.1 2007 Termination Competition .................. 95  
   8.2 2009 Termination Competition .................. 96  
   8.3 CS Processor Evaluation ...................... 97
<table>
<thead>
<tr>
<th>Contents</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8.4 Contributions of the CSDP Framework to mu-term</td>
<td>98</td>
</tr>
<tr>
<td>8.5 Advantages of the CSDP Framework over Previous Approaches</td>
<td>99</td>
</tr>
<tr>
<td><strong>Conclusions</strong></td>
<td>101</td>
</tr>
<tr>
<td><strong>Bibliography</strong></td>
<td>113</td>
</tr>
<tr>
<td><strong>Index</strong></td>
<td>127</td>
</tr>
<tr>
<td><strong>II Publications Associated to the Thesis</strong></td>
<td>129</td>
</tr>
<tr>
<td><strong>List of Publications</strong></td>
<td>131</td>
</tr>
<tr>
<td>Publications (full text)</td>
<td>133</td>
</tr>
<tr>
<td>8.6 Context-Sensitive Dependency Pairs</td>
<td>133</td>
</tr>
<tr>
<td>8.7 Improving the Context-Sensitive Dependency Graph</td>
<td>146</td>
</tr>
<tr>
<td>8.8 Proving Termination of Context-Sensitive Rewriting with MU-TERM</td>
<td>160</td>
</tr>
<tr>
<td>8.9 Usable Rules for Context-Sensitive Rewrite Systems</td>
<td>172</td>
</tr>
<tr>
<td>8.10 Improving Context-Sensitive Dependency Pairs</td>
<td>189</td>
</tr>
<tr>
<td>8.11 Context-Sensitive Dependency Pairs</td>
<td>206</td>
</tr>
<tr>
<td>8.12 Proving Termination in the CSDP Framework</td>
<td>254</td>
</tr>
<tr>
<td>8.13 Proving Termination Properties with MU-TERM</td>
<td>272</td>
</tr>
<tr>
<td><strong>Extended Versions</strong></td>
<td>281</td>
</tr>
<tr>
<td>8.14 Proving Termination in the CSDP Framework</td>
<td>281</td>
</tr>
</tbody>
</table>
List of Figures
1.1

Computing Wallis’ approximation to

. . . . . . . . . . . . . .

10

3.1
3.2

Delayed function call introduced by the rule a → c(f(a)) . . . .
The delayed function call is activated by the application of the
rule f(c(x)) → x . . . . . . . . . . . . . . . . . . . . . . . . . . .
The delayed function call b is introduced by the application of
the rule a → f(c(b)) . . . . . . . . . . . . . . . . . . . . . . . .
The context of the delayed function call b is modified by the
application of the rule f(x) → h(d(x)) . . . . . . . . . . . . . .

31

3.3
3.4

π
2

33
34
35

4.1

CSDPs of the CS-TRS in Example 10 . . . . . . . . . . . . . .

41

6.1
6.2
6.3
6.4
6.5

Estimated CS Dependency Graph for R in Example 10
. .
Estimated CS Dependency Graph for R in Example 9 . . . . .

60
61
63
64
77

Estimated CS Graph of Pairs from Example 52 following [AGL06]
Estimated CS Graph of Pairs from Example 52 following [AEF+ 08]

Estimated CS Dependency Graph from τ0 in Example 89 . . .


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


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 \cite{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 \cite{Thu10,ST00}. Rewriting techniques were used in mathematics and mathematical logic as a suitable framework to analyze computational properties as confluence and termination of λ-calculus and combinatory logic in the third and fourth decade of the 20th century \cite{Sch24,Chu32,CR36,New42}. Term rewriting is an operational principle that can be used in software engineering to implement and analyze abstract specifications \cite{Gut75}, as the foundational principle in the development of programming languages \cite{McC63,Han94,BJM00}, in automatically proving termination of programs \cite{GSSKT06,SKGAR07,AAC+08}, in equational unification \cite{Hul80}, etc. In other words, term rewriting has applications in practical computer science, theoretical computer science, and mathematics.

\section{Termination and Term Rewriting}

\textit{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 \textit{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 \textit{normal form} and is considered as the “result” of the Turing complete computational mechanism which we call \textit{term rewriting}.

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

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+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\(^1\), 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 \((F, R)\) 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 \((F, R)\) 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 \(R\) terminating, we have to find a reduction ordering which is *compatible* with the rules in \(R\) (i.e., \(\ell > r\) for each rule \(\ell \rightarrow r \in R\)). 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

---

\(^1\) 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\(^2\), 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\(^3\) \( g \). 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 \( u = F(l_1, \ldots, l_k) \) if \( \ell = f(l_1, \ldots, l_k) \), and where \( v = G(s_1, \ldots, s_m) \) if \( s = g(s_1, \ldots, s_m) \) 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 \( \ell \rightarrow r \in \mathcal{R} \).

---

\(^2\)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].

\(^3\)A symbol \( g \in \mathcal{F} \) is defined in \( \mathcal{R} \) if there is a rule in \( \mathcal{R} \) whose left-hand side is of the form \( g(l_1, \ldots, l_k) \).
1.1. Termination and Term Rewriting

Example 4
Continuing with Example 3, we have the following set of DPs DP(\(\mathcal{R}\)):

\[
\begin{align*}
F(f(x)) & \rightarrow F(g(f(x))) \\
F(f(x)) & \rightarrow F(x)
\end{align*}
\]

The rules in \(\mathcal{R}\) and the rules in 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:

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

Then, we have the following:

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

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

---

4If 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 \( u_1 \rightarrow v_1, u_2 \rightarrow v_2, \ldots \) 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
### 1.2. Context-Sensitive Rewriting

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:

\[
\begin{align*}
\text{if}(\text{true}, x, y) & \rightarrow x \\
\text{if}(\text{false}, x, y) & \rightarrow y
\end{align*}
\] (1.1) (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 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+07], OBJ2 [FGJM88], OBJ3 [GWM+00], CafeOBJ [FN97], etc., to improve the termination and efficiency of computations. Lazy languages (e.g., Haskell, Clean) interpret them as strictness annotations in order to become ‘more eager’ and efficient. Eager languages (e.g., Lisp, Maude, OBJ2, OBJ3, CafeOBJ) use them as replacement restrictions to become ‘more lazy’, thus (hopefully) avoiding nontermination.

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 \(R = (F, R)\), we also consider a replacement map \(\mu : F \rightarrow \wp(\mathbb{N})\) that satisfies \(\mu(f) \subseteq \{1, \ldots, k\}\), for each \(k\)-ary symbol \(f\) in the signature \(F\) [Luc98] (a pair \((R, \mu)\) 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 \(F\). In CSR we only rewrite \(\mu\)-replacing subterms: every term \(t\) (as a whole) is \(\mu\)-replacing by
1.2. Context-Sensitive Rewriting

definition; and \( t_i \) (as well as all its \( \mu \)-replacing subterms) is a \( \mu \)-replacing subterm of \( f(t_1, \ldots, t_k) \) if \( i \in \mu(f) \). In particular, if we fix \( \mu(if) = \{1\} \) 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 \( R \) [GM04, Example 49] provides a definition of integer division which advantageously uses CSR for handling if-then-else expressions:

\[
\begin{align*}
\text{if}(\text{true}, x, y) & \rightarrow x & (1.3) \\
\text{if}(\text{false}, x, y) & \rightarrow y & (1.4) \\
0 \geq s(y) & \rightarrow \text{false} & (1.5) \\
s(x) \geq s(y) & \rightarrow x \geq y & (1.6) \\
x \geq 0 & \rightarrow \text{true} & (1.7) \\
0 - y & \rightarrow 0 & (1.8) \\
s(x) - s(y) & \rightarrow x - y & (1.9) \\
0 \div s(y) & \rightarrow 0 & (1.10) \\
s(x) \div s(y) & \rightarrow \text{if}(x \geq y, s((x - y) \div s(y)), 0) & (1.11)
\end{align*}
\]

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 \( \mu(if) = \{1\} \) and \( \mu(f) = \{1, \ldots, \text{arity}(f)\} \) for all \( f \in F \setminus \{if\} \). 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 \( R \) and a replacement map \( \mu \), we say that \( R \) is \( \mu \)-terminating if no infinite \( \mu \)-rewrite sequence is possible.

**Example 10**

The TRS \( R \) in Figure 1.1 can be used to compute approximations to \( \frac{\pi}{2} \) by using Wallis’ product:

\[
\frac{\pi}{2} = \lim_{n \to \infty} \frac{2 \cdot 2 \cdot 4 \cdot 4 \cdots 2n \cdot 2n}{2n - 1 \cdot 2n + 1}
\]
evenNs → 0 : incr(oddNs)
oddNs → incr(evenNs)
incr(x : xs) → s(x) : incr(xs)
take(0, xs) → []
take(s(n), x : xs) → x :: take(n, xs)
zip([], xs) → []
zip(x, []) → []
zip(x : xs, y : ys) → (x ÷ y) : zip(xs, ys)
tail(x : xs) → xs
rep2([]) → []
rep2(x : xs) → x : (x : rep2(xs))
0 + n → n
s(n) + m → s(n + m)
0 * n → 0
s(n) * m → m + (n * m)
prodFrac(x ÷ y, z ÷ t) → (x * z) ÷ (y * t)
prodOffFracs([]) → s(0) ÷ s(0)
prodOffFracs(p :: ps) → prodFrac(p, prodOffFracs(ps))
halfPi(n) → prodOffFracs(take(n, zip(rep2(tail(evenNs)), tail(rep2(oddNs))))))
to an expression $\text{halfPi}(\sin(0))$ will obtain (disregarding the particular choice of such steps) an expression $\sin(0) ÷ \sin(0)$ representing the approximation $\frac{\pi}{2}$ to $\pi$, 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 [KKSV91, 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+04, DLM+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\(^5\) 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\(^6\).

**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

---

\(^5\)A TRS \(R\) is left-linear if no variable occurs twice in \(\ell\) for all \(\ell \rightarrow r \in R\).

\(^6\)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 $R'$ is the transformed TRS which is obtained from a TRS $R$ and a replacement map $\mu$, then $R$ is $\mu$-terminating whenever $R'$ 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-µ-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. According to this format a PhD can consist of a collection of papers previously published in relevant venues, together with an extended summary.
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. Introduction
Part I

Summary of the Research
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 $a_0 R a_1 R a_2 R a_3 R \cdots$. 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 $a_1 R a_2 R a_3 \cdots$. 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, if, from, true \ldots\}, 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 $T(\mathcal{F}, \mathcal{X})$, built from $\mathcal{F}$ and $\mathcal{X}$, is inductively defined as follows:

- $x \in T(\mathcal{F}, \mathcal{X})$ if $x \in \mathcal{X}$, and
- $f(t_1, \ldots, t_k) \in T(\mathcal{F}, \mathcal{X})$ if $t_1, \ldots, t_k \in T(\mathcal{F}, \mathcal{X})$, $f \in \mathcal{F}$ and $\text{ar}(f) = k$. 
The set of variables of a term \( t \in T(\mathcal{F}, \mathcal{X}) \) is denoted as \( \text{Var}(t) \). 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 \( \text{ar}(f) \) subtrees. Positions \( p, q, \ldots \) are chains of positive natural numbers that are used to address subterms of \( t \). The root position, referring to the whole term, corresponds to an empty chain and is denoted by \( \Lambda \). 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 \( C[] \in T(\mathcal{F} \cup \{ \Box \}, \mathcal{X}) \) with zero or more 'holes'. A hole \( \Box \) is a fresh constant symbol. We write \( C[]_p \) to denote that there is a hole \( \Box \) at position \( p \) of \( C[] \). Generally, \( C[] \) is written to denote an arbitrary context; we make the position of the hole explicit only if necessary. \( C[] = \Box \) is called the empty context.

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

### 2.3 Substitutions, Renamings, and Unifiers

A substitution is a mapping \( \sigma : \mathcal{X} \rightarrow T(\mathcal{F}, \mathcal{X}) \) where the set \( \text{Dom}(\sigma) = \{ x \in \mathcal{X} \mid \sigma(x) \neq x \} \) 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 \( \sigma(t(t_1, \ldots, t_k)) = t(\sigma(t_1), \ldots, \sigma(t_k)) \) for each \( k \)-ary function symbol \( f \in \mathcal{F} \) and terms \( t_1, \ldots, t_k \in T(\mathcal{F}, \mathcal{X}) \).

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 \( s, t \in T(\mathcal{F}, \mathcal{X}) \) is called a unifier of \( s \) and \( t \); it is also said that \( s \) and \( t \) unify (with substitution
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 \( s, t, t_1, \ldots, t_k \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \), we have \( f(t_1, \ldots, t_{i-1}, s, t_i+1, \ldots, t_k) R f(t_1, \ldots, t_{i-1}, t, t_i+1, \ldots, t_k) \) 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 \( \ell, r \in \mathcal{T}(\mathcal{F}, \mathcal{X}) \), \( \ell \not\in \mathcal{X} \) 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 \( \mathcal{F} = \mathcal{C} \cup \mathcal{D} \) 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 \( s \overset{p}{\rightarrow}_\mathcal{R} t \) (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 \( s \overset{q}{\rightarrow}_\mathcal{R} t \) if \( s \overset{q}{\rightarrow}_\mathcal{R} t \) 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 $s \overset{p}{\leadsto}_{\mathcal{R}, \mu} t$, $s \overset{p}{\leadsto}_{\mathcal{R}, \theta} t$, or $s \overset{p}{\leadsto} t$), if there is a nonvariable position $p \in \mathcal{P}os_{\mathcal{F}}(s)$ and a rule $t \overset{r}{\rightarrow} \in R$ (sharing no variable with $s$) such that $s|_{p}$ and $r$ 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 $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ [Luc98]. Let $M_{\mathcal{F}}$ be the set of all $\mathcal{F}$-maps (or $M_{\mathcal{R}}$ for the $\mathcal{F}$-maps of a TRS $\mathcal{R} = (\mathcal{F}, R)$). Let $\mu_{\top}$ be the replacement map given by $\mu_{\top}(f) = \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}$ (i.e., no replacement restrictions are specified).

A binary relation $R$ on terms is $\mu$-monotonic if for all symbols $f \in \mathcal{F}$, arguments $i \in \mu(f)$, and terms $s, t, t_{1}, \ldots, t_{k} \in T(\mathcal{F}, \mathcal{X})$, we have

$$f(t_{1}, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_{k}) \overset{f}{\underset{t_{i}}{\rightarrow}} f(t_{1}, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_{k})$$

whenever $s \overset{t}{\leadsto} f t$.

The set of $\mu$-replacing positions $\mathcal{P}os^{\mu}(t)$ of $t \in T(\mathcal{F}, \mathcal{X})$ is: $\mathcal{P}os^{\mu}(t) = \{\lambda\}$, if $t \not\in \mathcal{X}$ and $\mathcal{P}os^{\mu}(t) = \{\lambda\} \cup \bigcup_{i \in \mu(root(t))} i.\mathcal{P}os^{\mu}(t_{i})$, if $t \in \mathcal{X}$. 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 $\succeq_{\mu}$ is given by $s \succeq_{\mu} t$ if there is $p \in \mathcal{P}os^{\mu}(s)$ such that $t = s|_{p}$. We write $s \succeq_{\mu} t$ and say that $t$ is a strict $\mu$-replacing subterm of $s$ if $s \succeq_{\mu} t$ and $s \neq t$. We write $s \succeq_{\mu}^{c} t$ 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 $\mathcal{V}ar^{\mu}(t) = \{x \in \mathcal{V}ar(t) \mid t \succeq_{\mu} x\}$. The set of non-$\mu$-replacing variables of $t$, i.e., variables occurring at some frozen position in $t$, is $\mathcal{V}ar^{\#}(t) = \{x \in \mathcal{V}ar(t) \mid t \overset{\#}{\rightarrow} x\}$. 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: $s \overset{p}{\leadsto}_{\mathcal{R}, \mu} t$, written $s \overset{p}{\leadsto}_{\mathcal{R}} t$ (or $s \overset{p}{\leadsto}_{\mathcal{R}, \theta} t$, $s \overset{p}{\leadsto} t$, and even $s \overset{p}{\rightarrow} t$), if $s \overset{p}{\rightarrow} t$ 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):

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

Since the second argument of $(:)$ is non-$\mu$-replacing due to $\mu(:) = \{1\}$, $2.1 \not\in \mathcal{P}_{os}(0 : \text{incr(oddNs)})$. 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 $t = t_1 \rightarrow_{\mathcal{R}, \mu} t_2 \rightarrow_{\mathcal{R}, \mu} \cdots \rightarrow_{\mathcal{R}, \mu} t_n \rightarrow_{\mathcal{R}, \mu} \cdots$ starting from $t$. A TRS $\mathcal{R}$ is $\mu$-terminating if $\rightarrow_{\mu}$ is terminating.

A term $s$ $\mu$-narrows to a term $t$ (written $s \xrightarrow{p_{\mathcal{R}, \mu}} t$, $s \rightarrow_{\mathcal{R}, \mu, \theta} t$, or $s \rightarrow_{\mu, \theta} t$), if $s \xrightarrow{p} t$ and $p \in \mathcal{P}_{os}(s)$.
2. Preliminaries
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 $R = (C \cup D, R)$, the minimal nonterminating terms associated to $R$ are nonterminating terms $t$ whose proper subterms $u$ (i.e., $t \triangleright u$) are terminating. We denote as $T_\infty$ the set of minimal nonterminating terms associated to $R$ [HM04, HM07]. Considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term $t = f(t_1, \ldots, t_k) \in T_\infty$, 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., $t' \in T_\infty$; then

2. a rule $f(\ell_1, \ldots, \ell_k) \rightarrow r$ applies at the root of $t'$, i.e., $t' = \sigma(f(\ell_1, \ldots, \ell_k))$ for some substitution $\sigma$; furthermore, since $t' \in T_\infty$, $\sigma(x)$ is terminating for all $x \in \text{Var}(f(\ell_1, \ldots, \ell_k))$; and

3. there is a minimal nonterminating term $u \in T_\infty$ (hence, $\text{root}(u) \in D$) at some position $p$ of $\sigma(r)$ satisfying that $p \in \text{Pos}_F(r)$, (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 \( \mu \in M_F \), a simple extension of the notion of minimal term for unrestricted rewriting introduced above (i.e., \( T_{\infty} \)), is the following: let \( T_{\infty, \mu} \) be a set of minimal non-\( \mu \)-terminating terms in the following sense: \( t \) belongs to \( T_{\infty, \mu} \) 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 \( \mathcal{R} \):

\[
\begin{align*}
f(c(x)) & \rightarrow x \\
a & \rightarrow c(f(a))
\end{align*}
\]

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:

\[
f(a) \xrightarrow{\Lambda} R, \mu f(c(f(a))) \xleftarrow{\rightarrow R, \mu} f(a) \xrightarrow{\rightarrow R, \mu} \cdots
\]

Note that \( f(a) \xrightarrow{\Lambda} R, \mu f(c(f(a))) \) but \( f(c(f(a))) \notin T_{\infty, \mu} \) because \( f(c(f(a))) \triangleright f(a) \) and \( f(a) \) is not \( \mu \)-terminating. Then, minimality with respect to \( T_{\infty, \mu} \) is not preserved under inner \( \mu \)-rewritings.
Thus, $T_{\infty,\mu}$ 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 $R$ and a replacement map $\mu$, let $M_{\infty,\mu}$ be the set of minimal non-$\mu$-terminating terms in the following sense: $t$ belongs to $M_{\infty,\mu}$ if $t$ is non-$\mu$-terminating and every strict subterm $u$ at an active position (i.e., $t \not\Rightarrow_{\mu} u$) is $\mu$-terminating.

Note that $T_{\infty,\mu} \subseteq M_{\infty,\mu}$. Now, we have (see [AGL10, Section 3.1]):

1. Every non-$\mu$-terminating term $s$ contains a minimal non-$\mu$-terminating term $t \in M_{\infty,\mu}$ at an active position (i.e., $s \not\Rightarrow_{\mu} t$), and
2. minimal non-$\mu$-terminating terms $t$ are always rooted by a defined symbol $f \in D$: $\forall t \in M_{\infty,\mu}, \text{root}(t) \in D$.
3. Minimality is preserved under inner $\mu$-rewritings: for all $s \in M_{\infty,\mu}$, if $s \not\overset{\lambda}{\Rightarrow} t$ and $t$ is non-$\mu$-terminating, then $t \in M_{\infty,\mu}$.

Terms in $T_{\infty,\mu}$ 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:

$$f(a) \overset{R,\mu}{\Rightarrow} f(c(f(a))) \overset{R,\mu}{\Rightarrow} f(a) \overset{R,\mu}{\Rightarrow} \cdots$$

we have that $f(a), f(c(f(a))) \in M_{\infty,\mu}$, but only $f(a) \in T_{\infty,\mu}$.

### 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 $t = f(t_1, \ldots, t_k) \in M_{\infty,\mu}$. 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 $M_{\infty,\mu}$, i.e., $t' \in M_{\infty,\mu}$; then
2. a rule \( f(\ell_1, \ldots, \ell_k) \rightarrow r \) applies at the root of \( t' \) (i.e., \( t' = \sigma(f(\ell_1, \ldots, \ell_k)) \)) for some substitution \( \sigma \) such that \( \sigma(x) \) is \( \mu \)-terminating for all \( \mu \)-replacing variables \( x \in \text{Var}^\mu(f(\ell_1, \ldots, \ell_k)) \); and

3. one of the following holds:

   (a) There is a minimal non-\( \mu \)-terminating term \( u \in M_{\infty, \mu} \) at some position \( p \) of \( \sigma(r) \) satisfying that \( p \in \text{Pos}^\mu_r(r) \) (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 \( x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell) \) such that there is a minimal non-\( \mu \)-terminating term \( u \in M_{\infty, \mu} \) at some active position \( p \) of \( \sigma(x) \). That is, \( \sigma(x) = C[u]_p \) for some context \( C[\_]_p \) such that \( p \in \text{Pos}^\mu(C[\_]_p) \).

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 \( y \in \text{Var}^\mu(f(\ell_1, \ldots, \ell_k)) \), but we do not know whether or not \( \sigma(x) \) is \( \mu \)-terminating when \( x \in \text{Var}(f(\ell_1, \ldots, \ell_k)) \setminus \text{Var}^\mu(f(\ell_1, \ldots, \ell_k)) \). 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 \( R \) be a TRS and \( \mu \in M_R \). Then, for all \( t \in M_{\infty, \mu} \), there exist \( \ell \rightarrow r \in R \), a substitution \( \sigma \), and a term \( u \in M_{\infty, \mu} \) such that \( t \xrightarrow{\lambda} \sigma(\ell) \xrightarrow{\lambda} \sigma(r) \xrightarrow{\mu} u \) and either

1. there is a nonvariable \( \mu \)-replacing subterm \( s \) of \( r \), \( r \xrightarrow{\mu} s \), such that \( u = \sigma(s) \), or

2. there is \( x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell) \) such that \( \sigma(x) = C[u]_p \) for some context \( C[\_]_p \) with \( p \in \text{Pos}^\mu(C[\_]_p) \).

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

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

\[
t \xrightarrow{\lambda} \sigma_1(\ell_1) \xrightarrow{\lambda} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{\lambda} \sigma_2(\ell_2) \xrightarrow{\lambda} \sigma_2(r_2) \xrightarrow{\mu} t_2 \xrightarrow{\lambda} \cdots
\]
where, for all \( i \geq 1 \), \( \ell_i \rightarrow r_i \in \mathcal{R} \) are rewrite rules, \( \sigma_i \) are substitutions, and terms \( t_i \in M_{\infty, \mu} \) are minimal non-\( \mu \)-terminating terms such that either

1. \( t_i = \sigma_i(s_i) \) for some nonvariable term \( s_i \) such that \( r_i \sqsupseteq \mu s_i \), 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 \( C[\cdot]_{p_i} \) such that \( p_i \in \mathcal{P}os^\mu(C[\cdot]_{p_i}) \).

### 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 \( a \) is \( \mu \)-terminating. If we apply the rule \( a \rightarrow c(f(a)) \), we obtain \( c(f(a)) \), which contains the subterm \( f(a) \). Furthermore, \( f(a) \) 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 \( a \rightarrow c(f(a)) \)](image)

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 \( s, t \in T(\mathcal{F}, \mathcal{X}) \). If \( s \) is \( \mu \)-terminating, then:

1. If \( s \succeq_\mu t \), then \( t \) is \( \mu \)-terminating.
2. If \( s \rightharpoonup^*_{R,\mu} t \), 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 \( R \).

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

**Example 20** For \( R \) and \( \mu \) as in Example 9, the hidden terms of \( R \) 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 \( R \) 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 \( R \) and \( \mu \) as in Example 12, but instead of \( a \), consider \( f(a) \), which is minimal and non-\( \mu \)-terminating. If we apply the rule \( a \rightarrow c(f(a)) \) to this term, we obtain \( f(c(f(a))) \), which contains a context \( f(c(\Box)) \) that allows us to move the non-\( \mu \)-replacing and non-\( \mu \)-terminating term \( f(a) \) 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 \( a \rightarrow c(f(a)) \) 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 \( \sigma(x) = f(a) \).

**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 \( x, \sigma, C[^\mu p] \) and \( u \) as in Proposition 15 (2); then \( \sigma(x) = C[u]^\mu_p \) and \( u = \theta(u') \) for some hidden term \( u' \) and substitution \( \theta \) (see Theorem 28 below).
3.3. Hidden Terms

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 $R$:

\[
\begin{align*}
  a & \rightarrow f(c(b)) & h(x) & \rightarrow x \\
  f(x) & \rightarrow h(d(x)) & b & \rightarrow a
\end{align*}
\]

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

\[
\begin{align*}
  a & \mapsto_{R,\mu} f(c(b)) \mapsto_{R,\mu} h(d(c(b))) \mapsto_{R,\mu} d(c(b)) \mapsto_{R,\mu} d(c(a)) \mapsto_{R,\mu} \ldots
\end{align*}
\]

The hidden terms in the system are $c(b)$, $b$, and $d(x)$. When the rule $h(x) \rightarrow x$ is applied in the third $\mu$-rewriting step, we have that $\sigma(x) = C[u]_p = d(c(b))$ where $C[\square] = d(c(\square))$ and $u = b$. 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 $f$ contained in hidden terms $f(\ldots, r_i, \ldots)$ such that $r' \triangleright_{\mu} f(\ldots, r_i, \ldots) \triangleright_{\mu} r_i$ 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).
3. Structure of Infinite $\mu$-Rewrite Sequences

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

The symbols in this context conform the hiding context.

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

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

**Example 25** Continuing with the CS-TRS $(\mathcal{R}, \mu)$ in Example 9, we have that symbol $s$ 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 $\text{incr}$ hides position 1, symbol $\text{zip}$ hides positions 1 and 2, and symbol $\text{rep2}$ hides position 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 fresh variables that are not in $\text{Var}(t)$:

**Definition 26 (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

- $\text{REN}^\mu(f(t_1, \ldots, t_k)) = f([t_1]_f^{\mu}, \ldots, [t_k]_f^{\mu})$ for every $k$-ary symbol $f \in \mathcal{F}$, where given a term $s \in \mathcal{T}(\mathcal{F}, X)$, $[s]_i^f = \text{REN}^\mu(s)$ 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 $x_1, \ldots, x_n$.

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.

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
3. Structure of Infinite $\mu$-Rewrite Sequences

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 $t \in T(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X}$ 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 $\text{NARR}^{\mu}_{\mathcal{R}}(t)$ (or just $\text{NARR}^{\mu}(t)$) to indicate that $t$ is $\mu$-narrowable w.r.t. the (intended) TRS $\mathcal{R}$. We also let $\mathcal{NHT}(\mathcal{R}, \mu) = \{ t \in \mathcal{HT}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D} \text{ and } \text{NARR}^{\mu}_{\mathcal{R}}(\text{REN}^{\mu}(t)) \}$ 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 $t \in T_{\infty}^{\mu}$, there is an infinite sequence $t = t_0 \xrightarrow{\Lambda^{*}} \sigma_1(\ell_1) \xrightarrow{\Lambda}_{\mathcal{R}, \mu} \sigma_1(r_1) \supseteq_\mu t_1 \xrightarrow{\Lambda^{*}} \sigma_2(\ell_2) \xrightarrow{\Lambda}_{\mathcal{R}, \mu} \cdots$ where, for all $i \geq 1$, $\ell_i \rightarrow r_i \in R$ are rewrite rules, $\sigma_i$ are substitutions, and terms $t_i \in M_{\infty, \mu}$ 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 $x_i \in \text{Var}^{\mu}(r_i) \setminus \text{Var}^{\mu}(\ell_i)$, $t_i' \in \mathcal{NHT}(\mathcal{R}, \mu)$, 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\(^1\). 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 \( T_{\infty, \mu} \) and assuming \( \mu \)-narrowability of \( DHT \) thanks to [AGL10, Corol-
lary 2].

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

**Definition 29 (Hiding Context [AEF+08])** Let \((R, \mu)\) be a CS-TRS. The
function symbol \( f \) hides position \( i \) if there is a rule \( \ell \rightarrow r \in R \) with \( r \triangleright_\mu f(r_1, \ldots, r_i, \ldots, r_k) \), \( i \in \mu(f) \), 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
\( f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_k) \) where \( f \) hides position \( i \) and \( C' \) is a hiding con-
text.

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].

---

\(^1\) a symbol \( f \) is hidden if \( \text{root}(s) = f \) for some hidden term \( s \).
3. Structure of Infinite $\mu$-Rewrite Sequences
According to Theorem 28, an infinite minimal $\mu$-rewriting sequence whose starting term $t$ is strongly minimal has the following form:

$$t = t_0 \xrightarrow{\Lambda}^* \sigma_1(\ell_1) \xrightarrow{\Lambda_{R,\mu}} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{\Lambda}^* \sigma_2(\ell_2) \xrightarrow{\Lambda_{R,\mu}} \cdots$$

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., $t_{i-1} \xrightarrow{\Lambda}^* \sigma_i(\ell_i)$) 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., $r_i \xrightarrow{\mu} s_i$ and $\text{root}(s_i) \in D$), or

2. takes a minimal non-$\mu$-terminating and non-$\mu$-replacing subterm $t_i$ from $\sigma_i(\ell_i)$ (i.e., $\sigma_i(\ell_i) \xrightarrow{\mu} t_i$) 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 $t_i \in M_{\infty,\mu}$. 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 $\ell_{i+1}$ of a new rule $\ell_{i+1} \rightarrow r_{i+1}$, i.e., $t_i \xrightarrow{\Delta_\ast} \sigma_{i+1}(\ell_{i+1})$ 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:

- $\text{DP}_F(\mathcal{R}, \mu)$, which represents all possible direct ‘recursive’ calls in the very same sense of original DPs, and
- $\text{DP}_X(\mathcal{R}, \mu)$, 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 $t = f(t_1, \ldots, t_k) \in T(\mathcal{F}, \mathcal{X})$, we write $t^\sharp$ to denote the marked term $f^\sharp(t_1, \ldots, t_k)$.

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

\[
\begin{align*}
\text{DP}_F(\mathcal{R}, \mu) &= \{ t^\sharp \rightarrow s^\sharp \mid \ell \rightarrow r \in R, r \supseteq_\mu s, \text{root}(s) \in D, \ell \not\supseteq_\mu s, \text{narr}_\mu(\text{ren}_\mu(s)) \} \\
\text{DP}_X(\mathcal{R}, \mu) &= \{ t^\sharp \rightarrow x \mid \ell \rightarrow r \in R, x \in \text{Var}_\mu(r) \setminus \text{Var}_\mu(\ell) \}
\end{align*}
\]

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

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., $\ell \not\supseteq_\mu s$). 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.
4.1. Defining CSDPs

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 DP(R, µ) of CSDPs for R = (F, R) and µ 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 µ♯ as follows:

\[ \begin{align*}
\mu^\sharp (+^\sharp) &= \mu^\sharp (\text{TAKE}) = \mu^\sharp(\text{ZIP}) = \mu^\sharp (\ast^\sharp) = \mu^\sharp (\text{PRODFRAC}) = \{1, 2\} \\
\mu^\sharp (\text{HALFPI}) &= \mu^\sharp (\text{PRODOFFRACS}) = \mu^\sharp (\text{REP2}) = \mu^\sharp (\text{TAIL}) = \mu^\sharp (\text{INCR}) = \{1\} \\
\mu^\sharp (\text{EVENNS}) &= \mu^\sharp (\text{ODDNS}) = \emptyset \\
\mu^\sharp (f) &= \mu(f) \text{ for all } f \in F
\end{align*} \]

Figure 4.1: CSDPs of the CS-TRS in Example 10
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 \((R, \mu)\) 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 \(\text{IF}\) if the first argument is evaluated to `true` or `false`, respectively. Furthermore, we have the following noncollapsing pairs:

\[
s(x) \geq^t s(y) \rightarrow x \geq^t y \quad (4.24)
\]
\[
s(x) -^t s(y) \rightarrow x -^t y \quad (4.25)
\]
\[
s(x) \div^t s(y) \rightarrow x \geq^t y \quad (4.26)
\]
\[
s(x) \div^t s(y) \rightarrow \text{IF}(x \geq y, s((x - y) \div s(y)), 0) \quad (4.27)
\]

We define \(\mu^\#\) as:

\[
\mu^\#(\geq^t) = \mu^\#(-^t) = \mu^\#(\div^t) = \{1, 2\}
\]
\[
\mu^\#(\text{IF}) = \{1\}
\]
\[
\mu^\#(f) = \mu(f) \text{ for all } f \in \mathcal{F}
\]

A rule \(\ell \rightarrow r\) of a TRS \(\mathcal{R}\) is \(\mu\)-conservative if \(\text{Var}^\#(r) \subseteq \text{Var}^\#(\ell)\), 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 \([\text{AG00}, \text{GTSKF06}]\). 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 \(\text{DP}_\mathcal{F}(\mathcal{R}, \mu)\),
4.2. Unhiding TRS

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 $x_i \in \mathcal{V}_{\mu}(r_i) \setminus \mathcal{V}_{\mu}(\ell_i)$, we know that $\sigma(x_i) = \theta_i(C_i[\overline{t}_i])$ where $C_i[\cdot]$ is a hiding context and $\overline{t}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$ is a nonvariable hidden term such that $\theta(\overline{t}_i) \mu$-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 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 $C_i[\cdot]$ to extract the delayed call $t_i$, and then,
2. connect this delayed call, which is an instance $\theta(\overline{t}_i)$ of a hidden term $\overline{t}_i$, and the next CSDP.

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

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

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

- Once $t_i$ has been reached, we know that it is an instance $t_i = \theta(\overline{t}_i)$ of a nonvariable hidden term $\overline{t}_i \in \mathcal{NHT}(\mathcal{R}, \mu)$ 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:

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

**Definition 33 (Unhiding TRS [GL10a])** Let $\mathcal{R}$ be a TRS and $\mu \in M_\mathcal{R}$.

We define unh($\mathcal{R}, \mu$) as the TRS consisting of the following rules:

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

Then, the rules of the form \( f(x_1, \ldots, x_k) \rightarrow x_i \) simulate the \( \mu \)-replacing subterm relation for the hiding context and the rules \( t \rightarrow t^i \) simulate the marking of root symbols.

**Example 34**
Continuing with the CS-TRS \((R, \mu)\) in Example 10, we only have to consider the following rules to obtain the unhiding TRS:

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

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

\[
\begin{align*}
\text{incr} (x) & \rightarrow x \\
\text{rep2} (x) & \rightarrow x \\
\text{zip} (x, y) & \rightarrow x \\
\text{zip} (x, y) & \rightarrow y
\end{align*}
\]

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

\[
\begin{align*}
\text{incr}(\text{oddNs}) & \rightarrow \text{INCR}(\text{oddNs}) \\
\text{incr}(xs) & \rightarrow \text{INCR}(xs) \\
\text{oddNs} & \rightarrow \text{ODDNS} \\
\text{rep2}(xs) & \rightarrow \text{REP2}(xs) \\
\text{zip}(xs, ys) & \rightarrow \text{ZIP}(xs, ys)
\end{align*}
\]

**Example 35**
Continuing with the CS-TRS \((R, \mu)\) in Example 9, \( \text{unh}(R, \mu) \) is:

\[
\begin{align*}
x \div y & \rightarrow x & (4.28) \\
s(x) & \rightarrow x & (4.29) \\
(x - y) \div s(y) & \rightarrow (x - y) \div^i s(y) & (4.30) \\
x - y & \rightarrow x -^i y & (4.31)
\end{align*}
\]
4.3 Chains of CSDPs

As in the DP approach, the termination of CS-TRSs \((R, \mu)\) can be characterized as the absence of infinite (minimal) chains of CSDPs. Essentially, a chain of DPs is a (finite or infinite) sequence \(u_i \rightarrow v_i \in DP(R)\) together with a substitution \(\sigma\) such that \(\sigma(v_i) \rightarrow^* \sigma(u_{i+1})\) for all \(i \geq 1\), i.e., \(\sigma(v_i) \rightarrow^* \sigma(u_{i+1})\) 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 \(DP(R)\) by \(P\), where \(P\) 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 \(P\). Then, in order to model the behavior of collapsing pairs, we introduce a new TRS \(S\) which simulates the marking and the subterm relation by means of rewrite rules. In the following, given a TRS \(S\), we let \(S_{>\mu}\) be the rules \(s \rightarrow t \in S\) such that \(s \succ_{\mu} t\); and \(S_* = S \setminus S_{>\mu}\).

Remark 36 Note that, if we have an unhiding TRS \(unh(R, \mu)\), the rules in \(unh_{>\mu}(R, \mu)\) correspond with the rules of the form \(f(x_1, \ldots, x_i, \ldots, x_k) \rightarrow x_i\) and the rules in \(unh_{=} (R, \mu)\) with the rules of the form \(t \rightarrow t\).

Definition 37 (Chain of Pairs - Minimal Chain \([GL10a]\)) Let \(R, P\) and \(S\) be TRSs and \(\mu \in MR, P, S\). A \((P, R, S, \mu)\)-chain is a finite or infinite sequence of pairs \(u_i \rightarrow v_i \in P\), together with a substitution \(\sigma\) satisfying that, for all \(i \geq 1\),

1. if \(v_i \notin Var(u_i) \setminus Var^{\mu}(u_i)\), then \(\sigma(v_i) = t_i \leftarrow^*_{R, \mu} \sigma(u_{i+1})\), and

2. if \(v_i \in Var(u_i) \setminus Var^{\mu}(u_i)\), then \(\sigma(v_i) \leftarrow^*_{S_{>\mu}, \mu} \sigma(u_{i+1})\).

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

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

Theorem 38 (Characterization of \(\mu\)-Termination \([GL10a]\)) Let \(R\) be a TRS and \(\mu \in MR\). Let \(P = DP(R, \mu)\) and \(S = unh(R, \mu)\). Then, \(R\) is \(\mu\)-terminating if and only if there is no infinite minimal \((P, R, S, \mu)\)-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, \textsc{mu-term} was under revision in preparation for its participation in the 2006 International Termination Competition. The idea of adapting DPs to CSR came up again. The first preliminary definition of CSDPs was the following:

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

\[
\begin{align*}
\text{DP}_1(\mathcal{R}, \mu) &= \{\ell \rightarrow s | \ell \rightarrow r \in \mathcal{R}, r \geq_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\sim_\mu s\} \\
&\cup \{\ell \rightarrow \text{MUSUBTERM}(x) | \ell \rightarrow r \in \mathcal{R}, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(l)\} \\
&\cup \{\text{MUSUBTERM}(f(x_1, \ldots, x_k)) \rightarrow \text{MUSUBTERM}(x_i) | f \in \mathcal{F}, i \in \mu(f)\} \\
&\cup \{\text{MUSUBTERM}(f(x_1, \ldots, x_k)) \rightarrow f^\mu(x_1, \ldots, x_k) | f \in \mathcal{D}\}
\end{align*}
\]

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

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 \(s = f(s_1, \ldots, s_k)\) that (after marking its root symbol \(f\) as \(f^\mu\)) 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.
4.4. Historical Development of CSDPs

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 $(\mathcal{P}, \mathcal{R}, \mu^\triangledown)$-chain is a finite or infinite sequence of pairs $u_i \rightarrow v_i \in \mathcal{P}$, together with a substitution $\sigma$ satisfying that

$$\sigma(v_i) \leftarrow_{\mathcal{R}, \mu^\triangledown}^{\ast} \sigma(u_{i+1})$$

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 $\mathcal{R} = (\mathcal{C} \cup \mathcal{D}, R)$ 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:

$$\text{DP}_2,\mathcal{F}(\mathcal{R}, \mu) = \{ \ell \rightarrow s^\triangledown | \ell \rightarrow r \in \mathcal{R}, \mu \| s, \text{root}(s) \in \mathcal{D}, \ell \not\in \mu \}$$

$$\text{DP}_2,\mathcal{X}(\mathcal{R}, \mu) = \{ \ell \rightarrow U_{\ell, f, x}(x) | \ell \rightarrow r \in \mathcal{R}, f \in \mathcal{D}, x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(\ell) \}$$

$$\cup \{ U_{\ell, f, x}(f(x_1, \ldots, x_k)) \rightarrow f^\mu(x_1, \ldots, x_k) | \ell \rightarrow r \in \mathcal{R}, f \in \mathcal{D} \}$$

and $\mu^\triangledown(f) = \mu(f)$ if $f \in \mathcal{F}$, $\mu^\triangledown(f^\triangledown) = \mu(f)\upharpoonright \mathcal{D}$ if $f \in \mathcal{D}$, and $\mu^\triangledown(U_{\ell, f, x}) = \{1\}$ 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 $s = f(s_1, \ldots, s_k)$ 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 $(\mathcal{P}, \mathcal{R}, \mu^\triangledown)$-chain is a sequence of pairs $u_i \rightarrow v_i \in \mathcal{P}$ such that there is a substitution $\sigma$ satisfying either:

1. $\sigma(v_i) \leftarrow_{\mathcal{R}, \mu^\triangledown}^{\ast} \sigma(u_{i+1})$ if 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 \(x_i \in \mathcal{X}, w_i \in T(\mathcal{F}, \mathcal{X})\) and 
\[\sigma(x_i) \geq_\mu \sigma(w_i)\]
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 \(\mathcal{R} = (\mathcal{F}, \mathcal{R}) = (\mathcal{C} \cup \mathcal{T}, \mathcal{R})\) be a TRS, and let \(\mu \in \mathcal{M}_\mathcal{F}\). Let \(\text{DP}(\mathcal{R}, \mu) = \text{DP}_\mathcal{F}(\mathcal{R}, \mu) \cup \text{DP}_\mathcal{T}(\mathcal{R}, \mu)\) be the set of context-sensitive dependency pairs (CSDPs) where:

\[
\text{DP}_\mathcal{F}(\mathcal{R}, \mu) = \{ \ell \rightarrow s | \ell \rightarrow r \in \mathcal{R}, r \geq_\mu s, \text{root}(s) \in \mathcal{D}, \ell \not\in \mu, s, \}
\]

\[
\text{DP}_\mathcal{T}(\mathcal{R}, \mu) = \{ \ell \rightarrow x | \ell \rightarrow r \in \mathcal{R}, x \in \text{var}^\mu(r) \setminus \text{var}^\mu(\ell) \}
\]

We extend \(\mu \in \mathcal{M}_\mathcal{F}\) into \(\mu^\sharp \in \mathcal{M}_{\mathcal{F} \cup \mathcal{T}}\) by \(\mu^\sharp(f) = \mu(f)\) if \(f \in \mathcal{F}\), and \(\mu^\sharp(f^\sharp) = \mu(f)\) if \(f \in \mathcal{T}\).

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 \(\mu \in \mathcal{M}_\mu\). 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. \(\sigma(v_i) \leftarrow_{\mathcal{R}, \mu^\sharp} \sigma(u_{i+1})\), if \(u_i \rightarrow v_i \in \text{DP}_\mathcal{F}(\mathcal{R}, \mu)\), and

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

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 \(\text{prefix}_t(\Lambda) = \text{root}(t), \text{prefix}_t(i,p) = \text{root}(t).\text{prefix}_{i\mid p}(p)\). The strict prefix \(\text{sprefix}\) 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 \(\mathcal{R} = (\mathcal{F}, \mathcal{R})\) and \(\mathcal{P} = (\mathcal{G}, \mathcal{P})\) be TRSs and \(\mu \in \mathcal{M}_{\mathcal{F} \cup \mathcal{G}}\). 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 \(\sigma : \mathcal{X} \rightarrow T(\mathcal{F} \cup \mathcal{G}, \mathcal{X})\) satisfying that, for all \(i \geq 1\):
4.4. Historical Development of CSDPs

1. if \( v_i \notin \text{Var}(u_i) - \text{Var}_\mu(u_i) \), then \( \sigma(v_i) \leftarrow_{R_\mu} \sigma(u_{i+1}) \), 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 \( C_i \) such that \( p_i \in \text{Pos}^k(C_i) \), \( \text{sprefix}_{C_i} \) \( p_i \subseteq F \), and \( s_i^k \leftarrow_{R_\mu} \sigma(u_{i+1}) \).

A \((P, R, \mu)\)-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 \((R, \mu)\)-terminating, and

2. if \( v_i \in \text{Var}(u_i) - \text{Var}_\mu(u_i) \), then \( s_i^k \) is \((R, \mu)\)-terminating and \( \exists s_i \in \mathcal{NHT}(R, \mu) \) such that \( s_i = \sigma(s_i) \).

In [AEF+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+08, Definition 9] Let \( R \) be a TRS and \( \mu \in M_R \). If \( \text{DP}_X(R, \mu) \neq \emptyset \), we introduce a fresh unhiding tuple symbol \( U \) and the following unhiding DPs:

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

Let \( \text{DP}_u(R, \mu) \) be the set of all unhiding DPs (where \( \text{DP}_u(R, \mu) = \emptyset \) whenever \( \text{DP}_X(R, \mu) = \emptyset \)). Then \( \text{DP}'(R, \mu) = \text{DP}_X(R, \mu) \cup \text{DP}_u(R, \mu) \).

The corresponding definition of chain is, essentially, Definition 40 (but see [AEF+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:

\[
\begin{align*}
\text{TAIL}(x : xs) & \rightarrow U(xs) \\
\text{TAKE}(s(n), x : xs) & \rightarrow U(xs) \\
U(\text{incr}(x)) & \rightarrow U(x) \\
U(\text{incr}(\text{oddNs})) & \rightarrow \text{INCR}(\text{oddNs}) \\
U(\text{oddNs}) & \rightarrow \text{ODDNS}
\end{align*}
\]
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 un-hiding 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 $f(x_1, \ldots, x_k) \rightarrow x_i$ for all $f \in D$ and $i \in \mu(f)$ (where $x_1, \ldots, x_k$ are variables) are used in $\text{unh}_{\mu}(R, \mu)$, then we have the notion of minimal chain in [AGL10]; and if, additionally, rules $f(x_1, \ldots, x_k) \rightarrow F(x_1, \ldots, x_k)$ for all $f \in D$ are used in $\text{unh}_f(R, \mu)$, then we have the original notion of chain from [AGL06].
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 $\tau = (P, R, S, \mu)$ is a tuple, where $R$, $P$ and $S$ are TRSs and $\mu \in M_{R \cup P \cup S}$. The CS problem $(P, R, S, \mu)$ is finite if there is no infinite minimal $(P, R, S, \mu)$-chain. The CS problem $(P, R, S, \mu)$ is infinite if it is not finite or if $R$ is non-$\mu$-terminating.

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

**Definition 50 (CS Processor [AEF+08, AGL10])** A CS Processor is a function $\text{Proc}$ that takes a CS problem as an input and returns a new set of CS problems. Alternatively, $\text{Proc}$ can 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 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 $\tau = (P, R, S, \mu)$ and $P$ 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 $R$ be a TRS and $\mu \in M_R$. We construct a tree whose nodes are labeled with CS problems or “yes” or “no”, and whose root is labeled with $(\text{DP}(R, \mu), R, \text{unh}(R, \mu), \mu^\sharp)$. 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 $R$ 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 $R$ 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+08, AGL10], we gave the first definitions of CS problems and CS processors.

**Definition 53 (CS Problem [AEF+08, AGL10])** A CS problem \(\tau\) is a tuple \(\tau = (\mathcal{P}, \mathcal{R}, \mu)\), where \(\mathcal{R}\) and \(\mathcal{P}\) are TRSs and \(\mu \in M_{\mathcal{R}, \mathcal{P}}\). 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+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+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+08], the subterm condition (which is captured by the notion of \textit{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.
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 $(\gtrsim, \sqsubseteq)$ consists of a stable and monotonic quasi-ordering $\gtrsim$, and a stable and well-founded ordering $\sqsubseteq$ satisfying either $\gtrsim \circ \sqsubseteq \subseteq \sqsubseteq$ or $\sqsubseteq \circ \gtrsim \subseteq \sqsubseteq$ [KNT99]. Reduction pairs are used in the DP approach to witness the absence of infinite chains of (dependency) pairs by finding a reduction pair $(\gtrsim, \sqsubseteq)$ that is compatible with the rules and the DPs: $\ell \gtrsim r$ for all rewrite rules $\ell \rightarrow r$ and $u \sqsubseteq v$ 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 $u \sqsubseteq v$ (in this case, all other pairs $u \rightarrow v$ that are not strict must be compatible with the quasi-ordering $\gtrsim$, i.e., $u \gtrsim v$ must hold).

Stability is required both for $\gtrsim$ and $\sqsubseteq$ because, although we only check the left- and right-hand sides of the rewrite rules $\ell \rightarrow r$ (with $\gtrsim$) and pairs $u \rightarrow v$ (with $\gtrsim$ or $\sqsubseteq$), 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 $\sigma(\ell) \gtrsim \sigma(r)$ and also that $\sigma(u) \gtrsim \sigma(v)$ or $\sigma(u) \sqsubseteq \sigma(v)$.

Monotonicity is required for $\gtrsim$ 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 $\sqsubseteq$ (but, for this reason, we cannot compare the rules in $\mathcal{R}$ using $\sqsubseteq$). Endrullis et al. noted that transitivity is not necessary for the strict component $\sqsubseteq$ 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 (μ-Reduction Pair [AGL06])** Let $\mathcal{F}$ be a signature and $\mu \in M_\mathcal{F}$. A $\mu$-reduction pair $(\succeq, \sqsubseteq)$ consists of a stable and $\mu$-monotonic quasi-ordering $\succeq$ and a well-founded stable relation $\sqsubseteq$ on terms in $T(\mathcal{F}, X)$ which are compatible, i.e., $\succeq \circ \sqsubseteq \subseteq$ or $\sqsubseteq \circ \succeq \subseteq$.

We say that $(\succeq, \sqsubseteq)$ is $\mu$-monotonic if $\sqsubseteq$ 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 $(\succeq, \sqsubseteq)$ be a $\mu$-monotonic $\mu$-reduction pair such that $\ell (\succeq \cup \sqsubseteq) r$ for all $\ell \rightarrow r \in \mathcal{R}$. Let $\mathcal{R}_\sqsubseteq = \{ \ell \rightarrow r \in \mathcal{R} \mid \ell \sqsubseteq r \}$ and $\mathcal{S} = \mathcal{R} \setminus \mathcal{R}_\sqsubseteq$. 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):

\[
\mu(\text{if}) = \mu(\div) = \mu(\times) = \{1\} \quad \text{and} \quad \\
\mu(-) = \mu(\geq) = \mu(0) = \mu(\text{true}) = \mu(\text{false}) = \emptyset
\]

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

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

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:

\[
\begin{align*}
[\div](x, y) &= x + y \\
[\text{if}](x, y, z) &= x + y + z \\
[0] &= 0 \\
[\text{false}] &= 0 \\
[\geq](x, y) &= 0 \\
[-](x, y) &= x \\
[s](x) &= x + 1 \\
[\text{true}] &= 0
\end{align*}
\]
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 $S = a_1, a_2, \ldots, a_n, \ldots$ of objects $a_i$ belonging to a set $A$ can be represented as a path in a graph $G$ whose nodes are the objects in $A$, and whose arcs among them are appropriately established to represent $S$ (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 $S' = a_m, a_{m+1}, \ldots$ 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 $DG(\mathcal{R})$ is associated to the considered TRS $\mathcal{R}$. The nodes of the dependency graph are the DPs in $\mathcal{DP}(\mathcal{R})$; 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 $DG(\mathcal{P}, \mathcal{R})$ 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 (R, P, and S) together with a replacement map μ ∈ M_{R∪P∪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 R, P, and S be TRSs and μ ∈ M_{R∪P∪S}. The context-sensitive (CS) graph G(P, R, S, μ) has P as the set of nodes. Given u → v, u′ → v′ ∈ P, there is an arc from u → v to u′ → v′, if u → v, u′ → v′ is a minimal (P, R, S, μ)-chain for some substitution σ.

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

**Definition 58 (Context-Sensitive Dependency Graph [GL10a])** Let R be a TRS and μ ∈ M_R. The Context-Sensitive Dependency Graph (CSDG) for R and μ is DG(R, μ) = G(DP(R, μ), R, unh(R, μ), μ^♯).

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 τ = (P, R, S, μ) be a CS problem. Then, the processor Proc_{SCC} given by

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

is sound and complete, where S_Q are the rules from S involving a possible (Q, R, S, μ)-chain.

As a consequence of this theorem, we can work separately with the strongly connected components of G(P, R, S, μ), 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 → v and u′ → v′, it is undecidable whether they form a (P, R, S, μ)-chain. Then, for automation, an estimated CS graph is constructed. In the following, given a CS-TRS (P, μ) where P = (G, P),
6.2. Context-Sensitive (Dependency) Graph

we let $\mathcal{P}_X$ be the pairs $u \rightarrow v \in \mathcal{P}$ such that $v \in \text{Var}(u) \setminus \text{Var}(u)$; and $\mathcal{P}_G = \mathcal{P} \setminus \mathcal{P}_X$. When considering pairs $u \rightarrow v \in \mathcal{P}_G$, there is an arc from $u \rightarrow v$ to $u' \rightarrow v'$ if $\theta(v) \leftarrow \theta(u')$ for some substitution $\theta$. When considering collapsing pairs $u \rightarrow v \in \mathcal{P}_X$, we know that such pairs can only be followed by a pair $u' \rightarrow v' \in \mathcal{P}$ such that $\theta(t) \leftarrow \theta(u')$ for some $s \rightarrow t \in \mathcal{S}_t$ and substitution $\theta$. If $\mathcal{S}_G = \emptyset$, then there is no outcoming arc from any $u \rightarrow v \in \mathcal{P}_X$.

Following [GTSK05], we define $\text{TCAP}^\mu_R$ [AGL10, Subsection 8.2]. The idea is to obtain the maximal prefix context $C[\square]$ of $s$ (i.e., $s = C[s_1, \ldots, s_k]$ for some terms $s_1, \ldots, s_k$) that we know (without any ‘look-ahead’ for applicable rules) cannot be changed by any reduction starting from $s$. Furthermore, the above terms $s_1, \ldots, s_k$ must be rooted by defined symbols. Now, we replace those subterms $s_i$ that are at $\mu$-replacing positions (i.e., $s_i = s|_{p_i}$ for some $p_i \in \text{Pos}^\mu(s)$) 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 $\text{TCAP}^\mu_R$ be as follows:

$$
\text{TCAP}^\mu_R(x) = \begin{cases} 
  y & \text{if } x \text{ is a variable, and} \\
  f([t_1]^f_1, \ldots, [t_k]^f_k) & \text{if } f([t_1]^f_1, \ldots, [t_k]^f_k) \text{ does not unify with } \ell \text{ for any } \ell \rightarrow r \in \mathcal{R} \\
  y & \text{otherwise}
\end{cases}
$$

where $y$ is a fresh variable, $[s]^f_i = \text{TCAP}^\mu_R(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that $\ell$ shares no variable with $f([t_1]^f_1, \ldots, [t_k]^f_k)$ 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 $u \rightarrow v \in \mathcal{P}_G$ to $u' \rightarrow v' \in \mathcal{P}$ if $\text{TCAP}^\mu_R(v)$ and $u'$ unify, and

2. there is an arc from $u \rightarrow v \in \mathcal{P}_X$ to $u' \rightarrow v' \in \mathcal{P}$ if there is $s \rightarrow t \in \mathcal{S}_t$ such that $\text{TCAP}^\mu_R(t)$ 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 $\mu \in M_{\mathcal{R} \cup \mathcal{P} \cup \mathcal{S}}$. The estimated CS graph $\text{EG}(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ contains the CS graph $G(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$. 
Figure 6.1: Estimated CS Dependency Graph for \( \mathcal{R} \) in Example 10

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

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

where

- \( \mathcal{S}_Q = \emptyset \) if \( Q \chi = \emptyset \).
- \( \mathcal{S}_Q = \mathcal{S}_\mu \cup \{s \to t \mid s \to t \in \mathcal{S}_\mu, \text{TCAP}^\mu_{\mathcal{R}}(t) \text{ and } u' \text{ unify for some } u' \to v' \in Q\} \) if \( Q \chi \neq \emptyset \).

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 \( \text{TCAP}^\mu_{\mathcal{R}} \). 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\}\}
Example 65

For Example 9, we obtain the following CS problem

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

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:

\[
\text{Proc}_{\text{SCC}}(\tau) = \begin{cases} 
\{\{(4.24)\}, \mathcal{R}, \emptyset, \mu^\sharp\}, \\
\{\{(4.22), (4.23), (4.27)\}, \mathcal{R}, \text{unh}(\mathcal{R}, \mu) \setminus \{(4.31)\}, \mu^\sharp\}, \\
\{\{(4.25)\}, \mathcal{R}, \emptyset, \mu^\sharp\} 
\end{cases}
\]

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 \(\text{REN}\) and \(\text{CAP}\) in [AG00] to CSR.

Definition 66

Given a set \(\Delta\) of 'defined' symbols, we let \(\text{CAP}_\Delta^\mu\) be as follows:

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

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]:

\[ f(a, b, x) \rightarrow f(x, x, x) \]
\[ \begin{align*}
&c \rightarrow a \\
&c \rightarrow b
\end{align*} \]

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 \( \text{tcap} \) to CSR to obtain our latest over-approximation of the \( \text{cap} \) 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 \( R \) be a TRS and \( \mu \in \mathcal{M}_R \). The estimated CSDG consists of the set \( \text{DP}(R, \mu) \) of CSDPs together with arcs that connect them as follows:

1. There is an arc from a CSDP \( u \rightarrow v \in \text{DP}_R(R, \mu) \) to a CSDP \( u' \rightarrow v' \in \text{DP}_R(R, \mu) \) if \( \text{ren}^\mu(\text{cap}_R^\mu(v)) \) and \( u' \) unify.
2. There is an arc from a CSDP \( u \rightarrow v \in \text{DP}_X(R, \mu) \) to each CSDP \( u' \rightarrow v' \in \text{DP}_R(R, \mu) \).

For the TRS \( R \) 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 \( R \) be a TRS and \( \mu \in \mathcal{M}_R \). The CSDG consists of the set \( \text{DP}(R, \mu) \) of CSDPs together with arcs which connect them as follows:
1. There is an arc from a CSDP \( u \rightarrow v \in \text{DP}_F(R, \mu) \) to a CSDP \( u' \rightarrow v' \in \text{DP}(R, \mu) \) if \( \text{REN}^\mu(\text{CAP}^\mu(v)) \) and \( u' \) unify.

2. There is an arc from a CSDP \( u \rightarrow v \in \text{DP}_X(R, \mu) \) to a CSDP \( u' \rightarrow v' \in \text{DP}(R, \mu) \) if \( \text{root}(v') \in \mathcal{H}(R, \mu) \).

According to Definition 69, the estimated CSDG for the TRS \( R \) 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 \( R \) and \( P \) be TRSs and \( \mu \in M_{R \cup P} \). The estimated CS graph associated to \( R \) and \( P \) (denoted \( \text{EG}(P, R, \mu) \)) has \( P \) as the set of nodes and the arcs that connect them as follows:

1. There is an arc from \( u \rightarrow v \in \mathcal{P}_Y \) to \( u' \rightarrow v' \in \mathcal{P} \) if \( \text{TCAP}_R^\mu(v) \) and \( u' \) unify.

2. There is an arc from \( u \rightarrow v \in \mathcal{P}_X \) to \( u' \rightarrow v' \in \mathcal{P} \) if there is \( t \in \mathcal{NHT}(R, \mu) \) such that \( \text{TCAP}_R^\mu(t^\#) \) and \( u' \) unify.

According to Definition 70, the estimated CSDG for the TRS \( R \) in Example 52 is also depicted in Figure 6.1.

- The transformation of all collapsing pairs into noncollapsing ones as in [AEF+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+08].

![Diagram of CSDG](image)

Figure 6.4: Estimated CS Graph of Pairs from Example 52 following [AEF+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

$$\text{Proc}_{\text{Inf}}(\tau) = \begin{cases} \text{no} & \text{if } v = \theta(u) \\
\{ (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) \} & \text{for some } u \rightarrow v \in \mathcal{P}_0 \text{ and substitution } \theta; \text{ and otherwise} \end{cases}$$

is sound and complete.
Example 72

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

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

Together with $\mu(\text{id}) = \{1, 2\}$ and $\mu(\text{f}) = \{1, \ldots, \text{ar}(\text{f})\}$ for all other symbols $\text{f}$.

The set of CSDPs $\text{DP}(\mathcal{R}, \mu)$ is:

$$
\begin{align*}
\text{s}(x) \overset{x}{\rightarrow} \text{s}(y) & \rightarrow x \overset{x}{\rightarrow} y & (6.1) \\
x \overset{x}{\rightarrow} y & \rightarrow y \overset{x}{\rightarrow} 0 & (6.2) \\
x \overset{x}{\rightarrow} y & \rightarrow \text{IF}(y > 0, p(x) - p(y), x) & (6.3) \\
\text{s}(x) \overset{x}{\rightarrow} \text{s}(y) & \rightarrow (x - y) \overset{x}{\rightarrow} \text{s}(y) & (6.4) \\
\text{s}(x) \overset{x}{\rightarrow} \text{s}(y) & \rightarrow x \overset{x}{\rightarrow} y & (6.5) \\
\text{IF}(\text{false}, x, y) & \rightarrow y & (6.6) \\
x \overset{x}{\rightarrow} y & \rightarrow p(x) \overset{x}{\rightarrow} p(y) & (6.7)
\end{align*}
$$

Starting from $\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^2)$, the application of the processor $\text{Proc}_{\text{inf}}$ to $\tau$ yields “no” because for pair $(6.7)$, we have that $\theta(x' \overset{x}{\rightarrow} y') = p(x) \overset{x}{\rightarrow} p(y)$ using the substitution $\theta$ such that $\theta(x') = p(x)$ and $\theta(y') = p(y)$.

The following processor exploits an old observation about collapsing CSDPs in $[\text{AGL}06]$. The idea is to identify a subclass of collapsing pairs which do not generate infinite chains. Thus, the notion of $\text{DP}_X^1$ was defined:

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

When considering CS problems, we generalize this notion to deal with arbitrary pairs. We say that $f(u_1, \ldots, u_k) \rightarrow x \in \mathcal{P}_X^1$ if $f(u_1, \ldots, u_k) \rightarrow x \in \mathcal{P}_X$ and $x \in \text{Var}(u_i)$ for some $i \notin \mu(f)$.

The following CS processor allows us to conclude that a CS problem is finite if all its pairs are from $\mathcal{P}_X^1$.

Theorem 73 (Basic CS Processor for Collapsing Pairs [GL10a]) Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem where $\mathcal{R} = (\mathcal{C} \cup \mathcal{D}, R)$ and $\mathcal{S} = (\mathcal{H}, \mathcal{S})$. Assume that (1) all the rules in $\mathcal{S}_X$ are noncollapsing, i.e., for all $s \rightarrow t \in \mathcal{S}_X$, $t \notin X$ and (2) $\{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_X\} \cap \mathcal{D} = \emptyset$ and (3) for all $s \rightarrow t \in \mathcal{S}_X$, we have that $s = f(s_1, \ldots, s_k)$ and $t = g(s_1, \ldots, s_k)$ for some $k \in \mathbb{N}$, function symbols $f, g \in \mathcal{H}$, and terms $s_1, \ldots, s_k$. Then, the processor $\text{Proc}_{\text{fin}}$ given by

$$
\text{Proc}_{\text{fin}}(\tau) = \left\{ \begin{array}{ll}
\emptyset & \text{if } \mathcal{P} = \mathcal{P}_X^1 \\
\{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise}
\end{array} \right.
$$
is sound and complete.

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

**Example 74**
Consider the following TRS $R$ [Luc98, Example 15]:

$$
\begin{align*}
0 + x & \rightarrow x \\
s(x) + y & \rightarrow s(x + y) \\
\text{false} \land y & \rightarrow \text{false} \\
\text{true} \land x & \rightarrow x \\
\text{first}(0, x) & \rightarrow [] \\
\text{first}(s(x), y : z) & \rightarrow y : \text{first}(x, z) \\
\text{from}(x) & \rightarrow x : \text{from}(s(x)) \\
\text{if}(\text{false}, x, y) & \rightarrow y \\
\text{if}(\text{true}, x, y) & \rightarrow x
\end{align*}
$$

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

$$
\begin{align*}
0 + x & \rightarrow x \\
\text{true} \land x & \rightarrow x \\
\text{IF}(\text{false}, x, y) & \rightarrow y \\
\text{IF}(\text{true}, x, y) & \rightarrow x
\end{align*}
$$

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

$$
\begin{align*}
x + y & \rightarrow x + \sharp y \\
x + y & \rightarrow x \\
\text{first}(x, z) & \rightarrow \text{FIRST}(x, z) \\
\text{first}(x, z) & \rightarrow x \\
\text{first}(x, z) & \rightarrow z \\
\text{from}(s(x)) & \rightarrow \text{FROM}(s(x))
\end{align*}
$$

Let $τ = (\text{DP}(R, μ), R, \text{unh}(R, μ), μ^\sharp)$. Since $\text{Proc}_\text{Fin}(τ) = \emptyset$, we conclude that $τ$ 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+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 \( \mathcal{P} = (\mathcal{G}, \mathcal{P}) \) and, let \( \mathcal{P}_U \) be given by the following rules:

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

Here, \( U \) is a fresh symbol. Let \( \mathcal{P}' = (\mathcal{G} \cup \{U\}, \mathcal{P}') \) where \( \mathcal{P}' = (\mathcal{P}_X \setminus \mathcal{P}_X) \cup \mathcal{P}_U \), and \( \mu' \) extends \( \mu \) by \( \mu'(U) = \emptyset \). The processor \( \text{Proc}_{\text{Coll}} \) given by \( \text{Proc}_{\text{Coll}}(\tau) = \{(\mathcal{P}', \mathcal{R}, \emptyset, \mu')\} \) 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:

\[
\begin{align*}
\text{IF}(\text{false}, x, y) & \rightarrow U(y) \quad (6.8) \\
\text{IF}(\text{true}, x, y) & \rightarrow U(x) \quad (6.9) \\
U(s(x)) & \rightarrow U(x) \quad (6.10) \\
U(x \div y) & \rightarrow U(x) \quad (6.11) \\
U((x - y) \div s(y)) & \rightarrow (x - y) \div s(y) \quad (6.12)
\end{align*}
\]

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 $\text{MUSUBTERM}(f(x_1,\ldots,x_i,\ldots,x_k)) \rightarrow \text{MUSUBTERM}(x_i)$ for each function symbol $f$ in the signature and position $i \in \mu(f)$ and a pair $\text{MUSUBTERM}(f(x_1,\ldots,x_k)) \rightarrow F(x_1,\ldots,x_k)$ for each defined function symbol $f$. The transformation used in [AEF+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 $\text{Proc}_\text{Coll}$ 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 $P$ 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 $(\succcurlyeq, \sqsupseteq, \succeq)$ consists of a $\mu$-reduction pair $(\succcurlyeq, \sqsupseteq)$ and a stable relation on terms $\succeq$, which is compatible with $\succcurlyeq$ or $\sqsupseteq$, i.e., $\succeq \circ \succcurlyeq \subseteq \succcurlyeq$ or $\sqsupseteq \circ \succeq \subseteq \sqsupseteq$. We say that $(\succcurlyeq, \sqsupseteq, \succeq)$ is $\mu$-monotonic if $\sqsupseteq$ is $\mu$-monotonic.

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

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

1. $P \subseteq \succcurlyeq \cup \sqsupseteq$, $R \subseteq \succeq$, and
6. Reduction Triple Processor

2. whenever \( P_X \neq \emptyset \) we have that \( S \subseteq \geq \cup \sqsupset \cup \geq \).

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

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

is sound and complete.

These \( \mu \)-reduction triples can be used in combination with argument filterings, which discard subexpressions from constraints \( s \geq t \), \( s \sqsupset t \) or \( s \geq t \) in such a way that \( \pi(s) \geq \pi(t) \) (resp. \( \pi(s) \sqsupset \pi(t) \) or \( \pi(s) \geq \pi(t) \)) 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 \( i \in \{1, \ldots, k\} \) or a (possibly empty) list \([i_1, \ldots, i_m]\) of argument positions with \( 1 \leq i_1 < \cdots < i_m \leq k \) [KNT99].

Example 79

Continuing Example 65, for the CS problem

\( \tau_0 = \{(P_0, R, S_0, \mu_0^2)\} \)

where \( P_0 = \{(4.22), (4.23), (4.27)\} \) and \( S_0 = \text{unh}(R, \mu) \setminus \{(4.31)\} \), we iterate on \( \text{Proc}_{RT} \) as follows. With the following automatically generated polynomial interpretation:

\[
\begin{align*}
\text{[\text{\div}] (x, y)} &= x + 1 & \text{[\text{\geq}] (x, y)} &= x \\
\text{[if] (x, y, z)} &= x + y + z & \text{[\text{\neg}] (x, y)} &= 0 \\
\text{[0]} &= 0 & \text{[s] (x)} &= x \\
\text{[false]} &= 0 & \text{[true]} &= 0 \\
\text{[\text{\div}^\#] (x, y)} &= x + 1 & \text{[\text{IF}] (x, y, z)} &= x + y + z
\end{align*}
\]

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

\( \tau_1 = \{(P_0, R, S_1, \mu_1^2)\} \).

where \( S_1 = S_0 \setminus \{(4.28)\} \). For \( \tau_1 \), we obtain a new automatically generated polynomial interpretation:

\[
\begin{align*}
\text{[\text{\div}] (x, y)} &= x + y + 1 & \text{[\text{\geq}] (x, y)} &= x + y + 1 \\
\text{[if] (x, y, z)} &= y + z & \text{[\text{\neg}] (x, y)} &= x \\
\text{[0]} &= 0 & \text{[s] (x)} &= x + 1 \\
\text{[false]} &= 0 & \text{[true]} &= 1 \\
\text{[\text{\div}^\#] (x, y)} &= x + y + 1 & \text{[\text{IF}] (x, y, z)} &= y + z
\end{align*}
\]
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 \( S_1 \), thus leaving \( S_{2,\triangleright} \) empty:

\[
\tau_2 = \{ (P_0, R, S_2, \mu^i) \}.
\]

where \( S_2 = S_1 \setminus \{ (4.29) \} \). We can apply \( \text{Proc}_{RT} \) again over \( \tau_2 \) to remove the pair (4.27) with the following polynomial interpretation:

\[
\begin{align*}
\div (x, y) &= x + y + 1 \\
\text{if} (x, y, z) &= x + y + z + 1 \\
[0] &= 1 \\
\text{false} &= 0 \\
\div^* (x, y) &= x + y + 1 \\
\text{IF} (x, y, z) &= x + y + z
\end{align*}
\]

then we obtain the CS problem:

\[
\tau_3 = \{ (P_3, R, S_2, \mu^i) \}
\]

where \( P_3 = P_0 \setminus \{ (4.24) \} = P_{3,X}^c \). 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 \( u \to x \) 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 \( \succcurlyeq \) of reduction pairs \( (\succcurlyeq, \sqsupseteq) \) which are used in [AGL06, AGL10] are required to have the \( \mu \)-subterm property, i.e. \( \triangleright_\mu \subseteq \succcurlyeq \). This is equivalent to imposing \( f(x_1, \ldots, x_k) \succcurlyeq x_i \) for all projection rules \( f(x_1, \ldots, x_k) \to x_i \) with \( f \in \mathcal{F} \) and \( i \in \mu(f) \). This is similar for markings: in [AGL06] we have to ensure that \( f(x_1, \ldots, x_k) \succcurlyeq f^t(x_1, \ldots, x_k) \) for all defined symbols \( f \) in the signature. In [AGL10], thanks to the notion of hidden term, we relaxed the last condition: we require \( t \succeq t^d \) for all (narrowable) hidden terms \( t \). In [AEF+08], thanks to the notion of hiding context, we only require that \( \succcurlyeq \) be compatible with the projections \( f(x_1, \ldots, x_k) \to x_i \) for those symbols \( f \) and positions \( i \) such that \( f \) hides position \( i \). However, this information is implicitly encoded as (new) pairs \( \mathcal{U}(f(x_1, \ldots, x_k)) \to \mathcal{U}(x_i) \) in the set \( \mathcal{P} \). The strict component \( \sqsupseteq \) of the reduction pair \( (\succcurlyeq, \sqsupseteq) \) 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+08, AGL10]), we can relax the conditions
imposed on the orderings that deal with these rules. Furthermore, since rules in \( S \) 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 \( S \).

### 6.7 Reduction Triple Processor with Usable Rules

In order to use \( \text{Proc}_{RT} \), we require the rules in the TRS \( R \) of the CS problem \((P, R, S, \mu)\) to be included in \( \succeq \), i.e., \( R \subseteq \succeq \) 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 \( R \) 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 \( \text{rules}_R(f) = \{ \ell \rightarrow r \in R \mid \text{root}(\ell) = f \} \). The set of \( \mu \)-replacing symbols in a term \( t \in T(F, X) \) is denoted by \( \text{Fun}^{\mu}(t) = \{ f \mid \exists p \in \text{Pos}(t), f = \text{root}(t|_p) \} \). The simplest adaptation of this notion is the following.

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

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

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

\[
U^\uparrow(\tau) = \bigcup_{u \rightarrow v \in P, f \in \text{Fun}^{\mu}(v), f \uparrow_{R, \mu} g} \text{rules}_R(g)
\]

However, Definition 81 does not lead to a correct approach for proving termination of CSR.
Example 82

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

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

$$b \rightarrow c(b)$$

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:

$$b \rightarrow B$$

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

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

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

$$F(c(b), b) \rightsquigarrow_{\mu, \mu} F(b, b) \rightsquigarrow_{\mathcal{R}, \mu^\sharp} F(c(b), b) \rightsquigarrow_{\mu, \mu} \cdots$$

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 $\text{Var}^\mu(\ell) \cap \text{Var}^\mu(r) = \emptyset$; 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 CS, 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 \((P, R, S, \mu)\) where \(P\) is:

\[
\begin{align*}
G(a) & \rightarrow F(g(b)) \\
F(x) & \rightarrow x
\end{align*}
\]

the only rule in \(R\) is:

\[
b \rightarrow a
\]

and \(S\) 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 \(R\) is usable because it is necessary to build the following infinite minimal \((P, R, S, \mu)\)-chain:

\[
\begin{align*}
G(a) & \rightarrow_{P, \mu} F(g(b)) \rightarrow_{P, \mu} A \rightarrow_{S, \mu} G(b) & \rightarrow_{R, \mu} G(a) & \rightarrow_{P, \mu} \cdots
\end{align*}
\]

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 \((\mu\text{-conservative})\) non-\(\mu\)-terminating CS-TRS \(R\):

\[
\begin{align*}
a(x, y) & \rightarrow b(x, x) \\
d(x, e) & \rightarrow a(x, x) \\
b(x, g) & \rightarrow d(x, x) \\
g & \rightarrow e
\end{align*}
\]

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

\[
\begin{align*}
A(x, y) & \rightarrow B(x, x) \\
D(x, e) & \rightarrow A(x, x) \\
B(x, g) & \rightarrow D(x, x)
\end{align*}
\]

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

\[
\tau = (\text{DP}(R, \mu), R, \emptyset, \mu^*)
\]

is equivalent to \(\mu\)-termination of \(R\). 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 (\( \mathcal{C}_c \)) rules:
\[
\begin{align*}
    c(x, y) &\rightarrow x \\
    c(x, y) &\rightarrow y
\end{align*}
\]
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 the replacement restrictions now, the rule \( g \rightarrow e \) is not needed to capture the following infinite chain:
\[
A(g, g) \rightarrow_{\mu} B(g, g) \rightarrow_{\mu} D(g, g) \rightarrow_{\mathcal{R}} D(g, e) \rightarrow_{\mu} A(g, g) \rightarrow_{\mu} \cdots
\]

because we have the following sequence using \( \mathcal{C}_e \)-rules instead:
\[
A(c(g, e), c(g, e)) \rightarrow_{\mu} B(c(g, e), c(g, e)) \leftarrow_{\mathcal{C}_e} B(c(g, e), g) \rightarrow_{\mu} D(c(g, e), e) \leftarrow_{\mathcal{C}_e} D(c(g, e), e) \rightarrow_{\mu} A(c(g, e), c(g, e)) \rightleftharpoons_{\mu} \cdots
\]

However, if we consider the replacement restrictions now, the \( \mu \)-rewrite step
\[
B(c(g, e), c(g, e)) \leftarrow_{\mathcal{C}_e} B(c(g, e), g)
\]
is no longer possible. Since the infinite sequence above can be regarded as an infinite \( \mu \)-rewrite sequence:
\[
A(g, g) \rightarrow_{\mu} B(g, g) \rightarrow_{\mu} D(g, g) \leftarrow_{\mathcal{R}, \mu} D(g, e) \rightarrow_{\mu} A(g, g) \rightarrow_{\mu} \cdots
\]

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 \( t \in T(\mathcal{F}, \mathcal{X}) \) is denoted by \( \text{Fun}^f(t) = \{ f \mid \exists p \in \text{Pos}(t) \setminus \text{Pos}^\mu(t), f = \text{root}(t_p) \} \).

**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 \( g \in \text{Fun}^f(\ell) \cup \text{Fun}(r) \).

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 \( \mathcal{U}_\tau(\tau) \) of context-sensitive usable rules of \( \tau \) is
\[
\mathcal{U}_\tau(\tau) = \bigcup_{u \in \mathcal{P}, f \in \text{Fun}(u) \cup \text{Fun}(v), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_\mathcal{R}(g) \cup \bigcup_{\ell \rightarrow r \in \mathcal{R}, f \in \text{Fun}(\ell) \cup \text{Fun}(r), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_\mathcal{R}(g) \cup \bigcup_{s \rightarrow t \in \mathcal{S}, f \in \text{Fun}(s) \cup \text{Fun}(t), f \triangleright_{\mathcal{R}, \mu} g} \text{rules}_\mathcal{R}(g)
\]
6.7. Reduction Triple Processor with Usable Rules

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 $C_\varepsilon$-compatibility of $\supseteq$, i.e., $c(x, y) \supseteq x$ and $c(x, y) \supseteq y$ holds for a fresh function symbol $c$ [GL02a].

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

Let $\tau = (P, R, S, \mu)$ be a CS problem. Let $(\supseteq, \sqsubseteq, \supseteq)$ be a $\mu$-reduction triple such that

1. $P \subseteq \supseteq \cup \sqsubseteq$,
2. at least one of the following holds:
   - $U^\uparrow(\tau) \subseteq \supseteq$, $P \cup U^\uparrow(\tau)$ is strongly $\mu$-conservative, $\supseteq$ is $C_\varepsilon$-compatible
   - $U^\downarrow(\tau) \subseteq \supseteq$, $\supseteq$ is $C_\varepsilon$-compatible,
   - $R \subseteq \supseteq$,
3. and, whenever $P_X \neq \emptyset$, we have that $S \subseteq \supseteq \cup \sqsubseteq \cup \supseteq$.

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

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

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 $0 : s(0) : s(s(0)) : \cdots$ 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 ($zWquot$) function by means of rules in the following way:

$$
x - 0 \rightarrow 0 \quad (6.16)
$$
If our strategy is eager, the evaluation of any expression containing a call to
\texttt{from} starts an infinite computation. This is due to the recursive call to \texttt{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: \( \mu(:) = \{1\} \) and \( \mu(f) = \{1, \ldots, \text{ar}(f)\} \)
for all \( f \in \mathcal{F} \setminus \{:, \} \). Thanks to the replacement map, we avoid the following
DP:

\[ \text{from}(x) \rightarrow \text{FROM}(s(x)) \]

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

\[ \text{from}(s(x)) \rightarrow \text{FROM}(s(x)) \]
\[ \text{zWquot}(x, y) \rightarrow x \]
\[ \text{zWquot}(x, y) \rightarrow y \]

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

\[ \text{from}(s(x)) \rightarrow \text{FROM}(s(x)) \]
\[ \text{zWquot}(x, y) \rightarrow \text{ZwQUOT}(x, y) \]

We can now define the CS problem

\[ \tau_0 = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\downarrow) \]

The estimated CSDG \( \text{EG}(\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\downarrow) \) for \( \tau_0 \) is depicted in Figure 6.5. If we apply the SCC processor, we obtain:
6.7. Reduction Triple Processor with Usable Rules

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 $\tau_1 = \{(6.23), R, \emptyset, \mu_\rho\}$, $\tau_2 = \{(6.25), R, \emptyset, \mu_\rho\}$ and $\tau_3 = \{(6.26), R, \emptyset, \mu_\rho\}$.

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

$$\mathcal{U}^\uparrow(\tau_1) = \emptyset$$
$$\mathcal{U}^\downarrow(\tau_1) = \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\}$$

Since (6.23) is not strongly $\mu$-conservative, we use $\mathcal{U}^\uparrow(\tau_1)$ and the following polynomial interpretation:\footnote{The quasi-orderings $\succsim$ induced by a polynomial interpretation are always $\mathcal{C}_\varepsilon$-compatible.}

- $\mathsf{SEL}(x, y) = x + y$
- $\mathsf{div}(x, y) = x + y + 1$
- $\mathsf{int}(x, y) = y$
- $\mathsf{div}(x, y) = 0$
- $\mathsf{s}(x) = x + 1$

$\mathcal{U}^\uparrow(\tau_2) = \emptyset$
$$\mathcal{U}^\downarrow(\tau_2) = \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\}$$

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

- $\mathsf{min}(x, y) = x + y$
- $\mathsf{int}(x, y) = x + 1$

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

$$\mathcal{U}^\uparrow(\tau_3) = \{(6.16), (6.17)\}$$
$$\mathcal{U}^\downarrow(\tau_3) = \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\}$$

$\mathcal{U}^\uparrow(\tau_3) = \{(6.16), (6.17)\}$
$$\mathcal{U}^\downarrow(\tau_3) = \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\}$$

$\mathcal{U}^\uparrow(\tau_3) = \{(6.16), (6.17)\}$
$$\mathcal{U}^\downarrow(\tau_3) = \{(6.13), (6.16), (6.17), (6.18), (6.19), (6.20), (6.21), (6.22)\}$$
Since \{ (6.26) \} \cup \mathcal{U}(\tau_3) \text{ is strongly } \mu\text{-conservative, we can use the following polynomial interpretation:}
\[
\begin{align*}
\left(\frac{x}{y}\right)(x, y) &= x \\
[0] &= 0 \\
[s](x) &= x + 1
\end{align*}
\]
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) Thie mann 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 \(\mathcal R = (\mathcal F, \mathcal R)\) be a TRS and \(\Delta \subseteq \mathcal F\). Let \(>\) be an arbitrary total ordering over \(T(\mathcal F \cup \{\bot, c\}, \mathcal X)\) where \(\bot\) is a fresh constant symbol and \(c\) is a fresh binary symbol. The interpretation \(I_\Delta\) is a mapping from terminating terms in \(T(\mathcal F, \mathcal X)\) to terms in \(T(\mathcal F \cup \{\bot, c\}, \mathcal X)\) defined as follows:

\[
I_\Delta(t) = \begin{cases} 
  t & \text{if } t \in \mathcal X \\
  f(I_\Delta(t_1), \ldots, I_\Delta(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \Delta \\
  c(f(I_\Delta(t_1), \ldots, I_\Delta(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin \Delta
\end{cases}
\]
where \( t' = \text{order}(\{I_{\Delta}(u) \mid t \rightarrow_R u\}) \)

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

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 \( R = (F, R) \) be a TRS, \( \mu \in M_F \) and \( \Delta \subseteq F \). Let > be an arbitrary total ordering over \( T(F \cup \{\bot, c\}, \Delta) \) where \( \bot \) is a fresh constant symbol and \( c \) is a fresh binary symbol (with \( \mu(c) = \{1, 2\} \)). The \( \mu \)-interpretation \( I_{1,\mu} \) is a mapping from arbitrary terms in \( T(F, \Delta) \) to terms in \( T(F \cup \{\bot, c\}, \Delta) \) defined as follows:

\[
I_{1,\mu}(t) = \begin{cases} 
t & \text{if } t \in \Delta \\
f(I_{1,\mu}(t_1), \ldots, I_{1,\mu}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \Delta \\
c(f(I_{1,\mu}(t_1), \ldots, I_{1,\mu}(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin \Delta \text{ and } t \text{ is } \mu\text{-terminating}
\end{cases}
\]

where \( t' = \text{order}(\{I_{1,\mu}(u) \mid t \rightarrow_R \mu u\}) \)

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

- 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 \( R = (F, R) \) be a TRS, \( \mu \in M_F \) and \( \Delta \subseteq F \). Let > be an arbitrary total ordering over \( T(F \cup \{\bot, c\}, \Delta) \) where \( \bot \) is a fresh constant symbol and \( c \) is a fresh binary symbol. The basic \(\mu\)-interpretation \( I_{2,\mu} \) is a mapping from \(\mu\)-terminating terms in \( T(F, \Delta) \) to terms in \( T(F \cup \{\bot, c\}, \Delta) \) defined as follows:

\[
I_{2,\mu}(t) = \begin{cases} 
t & \text{if } t \in \Delta \\
f(I_{2,\mu, f, 1}(t_1), \ldots, I_{2,\mu, f, n}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in \Delta \\
c(f(I_{2,\mu, f, 1}(t_1), \ldots, I_{2,\mu, f, n}(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin \Delta
\end{cases}
\]

where \( I_{2,\mu, f, i}(t) = \begin{cases} 
I_{2,\mu}(t) & \text{if } i \in \mu(f) \\
t & \text{if } i \notin \mu(f)
\end{cases} \)

\[
\text{order}(T) = \begin{cases} 
\bot, & \text{if } T = \emptyset \\
c(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } >
\end{cases}
\]
In [AEF+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+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 \( R \) be a TRS. The set of root symbols associated to \( R \) is:

\[
\text{Root}(R) = \{ \text{root}(\ell) \mid \ell \to r \in R \} \cup \{ \text{root}(r) \mid \ell \to r \in R, r \notin X \}
\]

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

\[
\pi(t) = \begin{cases} t_{\pi(1)} & \text{if } t = f(t_1, \ldots, t_k) \text{ and } f \in \text{Root}(R) \\ t & \text{otherwise} \end{cases}
\]

**Theorem 96 (Subterm Processor [GL10b])** Let \( \tau = (P, R, S, \mu) \) be a CS problem where \( R = (F, R) = (C \uplus D, R) \), \( P = (G, P) \) and \( S = (H, S) \). Assume that (1) \( \text{Root}(P) \cap D = \emptyset \), (2) the rules in \( P_G \) are noncollapsing, and (3) if \( P_X \neq \emptyset \), for all \( s \to t \in S \), \( \text{root}(t) \in \text{Root}(P) \). Let \( P \) be a simple projection for \( P \). Let \( S_\pi = \{ s \to t \mid s \to t \in S \} \). Let \( P_{\pi,\triangleright,\mu} = \{ u \to v \in P \mid \pi(u) \triangleright,\mu \pi(v) \} \) and \( S_{\pi,\triangleright,\mu} = S_{\triangleright,\mu} \cup \{ s \to t \in S_\pi \mid s \triangleright,\mu \pi(t) \} \). Then, \( \text{Proc}_{\text{subterm}} \) given by

\[
\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} \{(P \setminus P_{\pi,\triangleright,\mu}, R, S \setminus S_{\pi,\triangleright,\mu}, \mu)\} & \text{if } \pi(P) \subseteq \triangleright,\mu \text{ and whenever } P_X \neq \emptyset, \text{ then } S_\pi \subseteq \triangleright,\mu \\ \{ (P, R, S, \mu) \} & \text{otherwise} \end{cases}
\]
6.8. Subterm Criterion

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.,

\[ \tau_1 = (\{(4.1)\}, \mathcal{R}, \emptyset, \mu^t) \]
\[ \tau_2 = (\{(4.14)\}, \mathcal{R}, \emptyset, \mu^s) \]
\[ \tau_3 = (\{(4.18)\}, \mathcal{R}, \emptyset, \mu^s) \]
\[ \tau_4 = (\{(4.19)\}, \mathcal{R}, \emptyset, \mu^s) \]

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

\[ \pi_1\left(\sum^{\#}\right) = 1 \]
\[ \pi_2\left(\#^{\ast}\right) = 1 \]
\[ \pi_3(\text{PRODOFFRACS}) = 1 \]
\[ \pi_4(\text{TAKE}) = 1 \]

that is:

\[ \pi_1(s(n) +^{\#} m) = s(n) \quad \triangleright_{\mu} \quad n = \pi_1(n +^{\#} m) \]
\[ \pi_2(s(n) *^{\#} m) = s(n) \quad \triangleright_{\mu} \quad n = \pi_2(n *^{\#} m) \]
\[ \pi_3(\text{PRODOFFRACS}(p::ps)) = p::ps \quad \triangleright_{\mu} \quad ps = \pi_3(\text{PRODOFFRACS}(ps)) \]
\[ \pi_4(\text{TAKE}(s(n), x:x)) = s(n) \quad \triangleright_{\mu} \quad n = \pi_4(\text{TAKE}(n,x)) \]

**Example 98**

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

\[ \tau_1 = (\{(4.24)\}, \mathcal{R}, \emptyset, \mu^s) \]
\[ \tau_2 = (\{(4.25)\}, \mathcal{R}, \emptyset, \mu^s) \]

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

\[ \pi_1\left(\geq^{\#}\right) = 1 \]
\[ \pi_2\left(\leq^{\#}\right) = 1 \]

that is:

\[ \pi_1(s(x) \geq^{\#} s(y)) = s(x) \quad \triangleright_{\mu} \quad x = \pi_1(x \geq^{\#} y) \]
\[ \pi_2(s(x) -^{\#} s(y)) = s(x) \quad \triangleright_{\mu} \quad x = \pi_2(x -^{\#} y) \]
Example 99

Consider \( R \) 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 \( \mu^\# \) in the usual way. The CSDPs \( \text{DP}(R, \mu) \) are:

\[
\begin{align*}
\text{s}(x) >^\# \text{s}(y) &\rightarrow x >^\# y \quad (6.29) \\
x -^\# y &\rightarrow y >^\# 0 \quad (6.30) \\
x -^\# y &\rightarrow \text{IF}(y > 0, p(x) - p(y), x) \quad (6.31) \\
\text{s}(x) \div^\# \text{s}(y) &\rightarrow (x - y) \div^\# \text{s}(y) \quad (6.32) \\
\text{s}(x) \div^\# \text{s}(y) &\rightarrow x -^\# y \quad (6.33) \\
\text{IF}(\text{true}, x, y) &\rightarrow x \quad (6.34) \\
\text{IF}(\text{false}, x, y) &\rightarrow y \quad (6.35)
\end{align*}
\]

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

\[
\begin{align*}
x - y &\rightarrow x \quad (6.36) \\
x - y &\rightarrow y \quad (6.37) \\
p(x) - p(y) &\rightarrow p(x) -^\# p(y) \quad (6.38) \\
p(x) &\rightarrow P(x) \quad (6.39)
\end{align*}
\]

If we consider the CS problem \( \tau_0 = (\text{DP}(R, \mu), R, \text{unh}(R, \mu), \mu^\#) \) and we apply \( \text{Proc}_{\text{SCC}}(\tau_0) \), one of the resulting problems is \( \tau_1 = (P_1, R, S_1, \mu^\#) \), where \( P_1 = \{(6.34), (6.35), (6.31)\} \) and \( S_1 = \{(6.38), (6.36), (6.37)\} \). By applying \( \text{Proc}_{\text{UR}}(\tau_1) \), we can remove the pair (6.34) with the following polynomial interpretation:

\[
\begin{align*}
[-^\#](x, y) &= 2x + 2y + \frac{1}{2} \\
[-](x, y) &= 2x + 2y + \frac{1}{2} \\
[>] (x, y) &= 2x + \frac{1}{2} y \\
\text{if}(x, y, z) &= \frac{1}{2} x + y + z \\
\text{p}(x) &= \frac{1}{2} x \\
\text{true} &= 2 \\
0 &= 0 \\
\text{s}(x) &= 2x + 2
\end{align*}
\]

where all the rules are usable except the (\( \div^\# \))-rules. The resulting CS problem is \( \tau_2 = (P_2, R, S_1, \mu^\#) \) where \( P_2 = \{(6.35), (6.31)\} \). By applying the subterm criterion, we can remove all rules in \( S_1 \) with the following projection for \( P \):

\[
\begin{align*}
\pi(\text{IF}) &= 3 \\
\pi(\ -^\#) &= 1
\end{align*}
\]
that is:
\[
\begin{align*}
\pi(\text{IF}(\text{false}, x, y)) &= y \geq_{\mu} y = \pi(y) \\
\pi(x - t y) &= x \geq_{\mu} x = \pi(\text{IF}(y > 0, p(x) - p(y), x)) \\
\pi(p(x) - p(y)) &= p(x) - p(y) \triangleright_{\mu} p(x) = \pi(p(x) - t p(y)) \\
\pi(p(x) - p(y)) &= p(x) - p(y) \triangleright_{\mu} x = \pi(x) \\
\pi(p(x) - p(y)) &= p(x) - p(y) \triangleright_{\mu} y = \pi(y)
\end{align*}
\]

And obtaining the CS problem:
\[
\tau_3 = (\{(6.35), (6.31)\}, R, \emptyset, \mu^t)
\]

which is proved to be finite by using\(^2\)\(\text{Proc}_{SCC}(\tau_3)\).

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 \(R\), but only the rules in \(P \cup S\).

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

1. for all \(f \in \text{Root}(P)\), \(\pi(f) \notin \mu(f)\),
2. \(\pi(P) \subseteq \succ\), and,
3. whenever \(S \neq \emptyset\) and \(P_X \neq \emptyset\), we have that \(S_{\pi} \subseteq \succ\)

Let \(P_{\pi, >} = \{u \rightarrow v \in P \mid \pi(u) > \pi(v)\}\) and \(S_{\pi, >} = \{s \rightarrow t \in S_{\triangleright_{\mu}} \mid s > t\} \cup \{s \rightarrow t \in S_G \mid s > \pi(t)\}\). Then, the processor \(\text{Proc}_{NRP}\) given by
\[
\text{Proc}_{NRP}(P, R, S, \mu) = \begin{cases} 
\{(P \setminus P_{\pi, >}, R, S \setminus S_{\pi, >}, \mu)\} & \text{if (1), (2), and (3) hold} \\
\{(P, R, \mu)\} & \text{otherwise}
\end{cases}
\]
is sound and complete.

\(^2\)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/cardiv.trs).
Example 101

Consider the following TRS \( R \) [Zan97, Example 1]:

\[
\begin{align*}
g(x) & \rightarrow h(x) \\
c & \rightarrow d \\
h(d) & \rightarrow g(c)
\end{align*}
\]

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

\[
\begin{align*}
G(x) & \rightarrow H(x) \\
H(d) & \rightarrow G(c)
\end{align*}
\]

and \( unh(R, \mu) \) is:

\[
c \rightarrow C
\]

Then, for the CS problem \( \tau = (DP(R, \mu), R, unh(R, \mu), \mu^d) \), we can apply \( Proc_{NRP} \) to remove the second pair in \( \tau \) by using the following projection\(^3\):

\[
\begin{align*}
\pi(G) &= 1 \\
\pi(H) &= 1
\end{align*}
\]

and the following polynomial interpretation:

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

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 \( \mathcal{C} \) including noncollapsing pairs only, i.e., \( \mathcal{C} \subseteq DP_F(R, \mu) \). A similar technique based on using stable quasi-orderings to compare pairs (but without paying attention to the rules in \( R \)
as in the subterm criterion) was introduced and proved correct under certain conditions. In [AGL10], these techniques were defined as processors. A new

\(^3\)Since \( DP_X(R, \mu) = \emptyset \), 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).
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\(^1\) 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+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

\(^1\)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.

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.

\[
\text{data} \ \text{Term} \ \text{idF} \ \text{idV} = \text{F} \ \text{idF} \ [\text{Term} \ \text{idF} \ \text{idV}] \\
| \ \text{V} \ \text{idV}
\]

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.

\[
\text{class HasArity a where} \\
\quad \text{getFArity :: a -> Int} \\
\quad \text{setFArity :: Int -> a -> a}
\]

\[
\text{class HasMu a where} \\
\quad \text{getFMu :: a -> [Int]} \\
\quad \text{setFMu :: [Int] -> a -> a}
\]

\(^2\)http://www.haskell.org/
With this information, we can define the different kind of systems which are handled by \textsc{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:

\begin{verbatim}
data IdFTRS = IdFTRS
  { fid :: Int
  , fname :: String
  , arity :: Int
  }
\end{verbatim}

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

\begin{verbatim}
instance HasArity IdFTRS where
  getFArity = arity
  setFArity ar (IdFTRS { fid = fid'
                         , fname = fname'
                         }) = IdFTRS { fid = fid'
                                            , fname = fname'
                                            , arity = ar
                         }
\end{verbatim}

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

\begin{verbatim}
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 = csprev
                       , mu = csmu })
\end{verbatim}
= IdFCS { csfprev = csprev \\
    , mu = newmu \\
} 

Then, the type of our context-sensitive function symbol is \texttt{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 \( P, R, \) and \( S \) (the information about the replacement map is encoded as part of the description of the identifiers). As mentioned in Section 7.2, \texttt{mu-term} implements different frameworks (DP framework, CSDP framework, A\textsc{v}C framework...) and we have different kinds of problems. A CS problem is defined in the following way:

```plaintext
data CS = CS

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

To create CS problems or to modify their information, we use a different typeclass:

```plaintext
instance MkCSProblem CS trs where 
    mkCSProblem CS r p s = CSProb r p s 
    mapR f (CSProb r p s) = CSProb (f r) p s 
    mapP f (CSProb r p s) = CSProb r (f p) s 
    mapS f (CSProb r p s) = CSProb r p (f s)
```
Note that when we treat CS problems, the variable \(\text{trs}\) is instantiated to \(\text{TRS idF idV (Term idF idV)}\) where the identifier \(\text{idF}\) is \(\text{IdFCS IdFTRS}\), which is an instance of \(\text{HasArity}\) and \(\text{HasMu}\).

### 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`).

```haskell
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.

```haskell
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 $P_\lambda^1$ 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.

```haskell
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:

```haskell
-- Processor
instance (trs ~ TRS idF idV (Term idF idV)
            , HasArity 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 $P_1^1$; 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 TTT2 [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)
94 7. Implementing the CSDP Framework

-> 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 lan-
guage 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 dif-
ferent 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 re-
turns 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.
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 [AGLN10] of our CSDP framework has improved the state of the art of termination of CSR and the International Termination Competition 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 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:


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

1http://www.lri.fr/~marche/termination-competition/
2http://www.lri.fr/~marche/tpdb
Problem Data Base (TPDB, version 4.0). The results are summarized in Table 8.1.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Proved</th>
<th>YES Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>mu-term 4.4</td>
<td>68/90</td>
<td>2.87s</td>
</tr>
<tr>
<td>AProVE 07</td>
<td>64/90</td>
<td>6.90s</td>
</tr>
</tbody>
</table>

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 \((R, \mu)\). 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 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+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.

---

3See http://termcompuibk.ac.at/
### 8.3 CS Processor Evaluation

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.

<table>
<thead>
<tr>
<th>Tool Version</th>
<th>Proved</th>
<th>Average time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AProVE</td>
<td>34/37</td>
<td>3.084s</td>
</tr>
<tr>
<td>Jambox</td>
<td>28/37</td>
<td>2.292s</td>
</tr>
<tr>
<td>MU-TERM</td>
<td>34/37</td>
<td>1.277s</td>
</tr>
<tr>
<td>VMTL</td>
<td>29/37</td>
<td>6.708s</td>
</tr>
</tbody>
</table>

Table 8.2: 2009 International Termination Competition results

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 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 \( N_2 = \{0, 1, 2\} \).**

Table 8.3: Summary of processors used in MU-TERM
(d) **RT processor with LPoly and coefficients in** $Q_2 = \{0, 1, 2, \frac{1}{2}\}$ and $Q_4 = \{0, 1, 2, 3, 4, \frac{1}{2}, \frac{3}{2}\}$ (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\(^4\), where we compare the first version of MU-TERM that uses CSDPs in 2006 [AGL06] (MU-TERM 4.3),

\(^4\)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 [AGLMN10] (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.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Proved</th>
<th>YES Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MU-TERM 4.3</td>
<td>65/109</td>
<td>3.37s</td>
</tr>
<tr>
<td>MU-TERM 4.4</td>
<td>80/109</td>
<td>1.07s</td>
</tr>
<tr>
<td>MU-TERM 5.0</td>
<td>95/109</td>
<td>1.13s</td>
</tr>
</tbody>
</table>

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+08].

```maude
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 NatList -> 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(M, L)) = s(length(L)).
endfm
```
We can use the transformation developed in [DLM+08] to transform this system into a CS-TRS (without sorts). Such a CS-TRS can be found in the Termination Problems Data Base\(^5\) (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\(^6\).

\(^5\)http://www.lri.fr/~marche/tpdb/
\(^6\)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 $R$.

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 $S$ 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 Sterngagel [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 [AGLM10]). 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 cadenas minimales, 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 desembozan 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 \( S \) 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 \textsc{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-μ-terminants són crusials per a entendre el comportament no-μ-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 sensibles 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 cadenes minimales, 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 crusials 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 minimals en les seqüències de μ-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 μ-terminació de $R$.

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 demanen 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 competència 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 provenen de la reescritura sensible al context.

Com a treball futur estem interessats a abordar les següents tasques i problemes:

- Aprofitar la presència del component $S$ en la definició de problema sensible al context per a definir nous processadors que tinguen 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 automats 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 reescritura sensible al context usant parells de dependència sensibles al context. La modularitat de la reescritura sensible al context va ser investigada per primera vegada en [GL02a, GL02b]. El nostre pla és considerar els mòduls de reescritura de Urbain [Urb04] i la seua anàlisi de la modularitat de la terminació per a estendre els seus resultats a la reescritura sensible al context.

- Provar no-terminació de la reescritura 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 MÚ-TERM amb un processador per a la detecció de no-terminació de la reescritura sensible al context més potent (veure [AGLNM10]). Planifiquem usar la nostra experiència en l’anàlisi de les seqüències de $\mu$-reescritura 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 apropides ni definicions de Coq/Isabelle de tècniques específiques per a provar la terminació de la reescriptura sensible al context.
Bibliography


[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, 


[GTSK05] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In B. Gramlich, editor, 

[GTSKF04] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Automated Termination Proofs with AProVE. In V. van Oostrom, editor, 


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
cor- sensitivity 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


8.6 Context-Sensitive Dependency Pairs

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 $\mu : F \rightarrow \mathcal{P}(\mathbb{N})$ satisfying $\mu(f) \subseteq \{1, \ldots, k\}$, for each $k$-ary symbol $f$ of a signature $F$ [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 $f(t_1, \ldots, t_k)$ 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 $R$ (i.e., for proving the $\mu$-termination of $R$). 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-O04-02, 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.

Context-Sensitive Dependency Pairs

[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 \( R \), the dependency pairs associated to \( R \) conform a new TRS \( DP(R) \) which (together with \( R \)) determines the so-called dependency chains whose finiteness or infiniteness characterize termination of \( R \).

Given a rewrite rule \( l \rightarrow r \), we get dependency pairs \( l \rightarrow s \) for all subterms \( s \) of \( r \) which are rooted by a defined symbol; the notation \( t \) means that the root symbol \( f \) of \( t \) is marked thus becoming \( f^\# \) (often just capitalized: \( F \)). A chain of dependency pairs is a sequence \( u_i \rightarrow v_i \) of dependency pairs such that \( \sigma(v_i) \) rewrite 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 \( R \) [GM99, Example 1]:

\[
\begin{align*}
c & \rightarrow a \\
f(a,b,X) & \rightarrow f(X,X,X) \\
c & \rightarrow b
\end{align*}
\]

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 \( R^\mu_C \)) could be used in this case, but no concrete proof of termination for \( R^\mu_C \) is known yet. Furthermore, \( R^\mu_C \) has 13 dependency pairs and the dependency graph contains many cycles. In contrast, \( R \) 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 \( R \) 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 \( R \):

\[
\begin{align*}
a & \rightarrow c(f(a)) \\
f(c(X)) & \rightarrow X
\end{align*}
\]

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 \( R \) is \( \mu \)-terminating, which is not true:

\[A \text{ symbol } f \text{ is said to be defined in a TRS } R \text{ if } R \text{ contains a rule } f(l_1, \ldots, l_k) \rightarrow r.\]
Indeed, we must add the following dependency pair
\[ F(\text{c}(X)) \rightarrow X \]
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 characterize termination of CSR. We also show how to use term orderings for proving termination of CSR by means of the new approach. Furthermore, we are properly extending Arts and Giesl’s approach: whenever \( \mu(f) = \{1, \ldots, k\} \) for all \( k \)-ary symbols \( f \in F \), 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, \( X \) denotes a countable set of variables and \( F \) denotes a signature, i.e., a set of function symbols \( \{f, g, \ldots\} \), each having a fixed arity given by a mapping \( \text{ar} : F \rightarrow N \). The set of terms built from \( F \) and \( X \) is \( T(F, X) \).

Positions \( p, q, \ldots \) are represented by chains of positive natural numbers used to address subterms of \( t \). Given positions \( p, q \), we denote their concatenation as \( p.q \).

If \( p \) is a position, and \( Q \) is a set of positions, \( p.Q = \{p.q \mid q \in Q\} \). We denote the topmost position by \( \Lambda \). The set of positions of a term \( t \) is \( \text{Pos}(t) \).

Positions of non-nullable symbols in \( t \) are denoted as \( \text{Pos}_X(t) \), and \( \text{Pos}_X(t) \) are the positions of variables. The subterm at position \( p \) of \( t \) is denoted as \( t|_p \), and \( t|_p.s \) is the term \( t \) with the subterm at position \( p \) replaced by \( s \). We write \( t \sqsupset s \) if \( s = t|_p \) for some \( p \in \text{Pos}(t) \) and \( t \) is a subterm of \( t \).

The symbol labelling the root of \( t \) is denoted as \( \text{root}(t) \). A context is a term \( C \in T(F \cup \{\square\}, X) \) with zero or more ‘holes’ \( \square \) (a fresh constant symbol).

A rewrite rule is an ordered pair \( (l, r) \), written \( l \rightarrow r \), with \( l, r \in T(F, X) \), \( l \not\in X \) 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 \( R = (F, R) \) where \( R \) is a set of rewrite rules. Given \( R = (F, R) \), we consider \( F \) as the disjoint union \( F = C \cup D \) of symbols \( c \in C \), called constructors and symbols \( f \in D \), called defined functions, where \( D = \{\text{root}(l) \mid l \rightarrow r \in R\} \) and \( C = F - D \).

Context-sensitive rewriting. A mapping \( \mu : F \rightarrow \mathcal{P}(\mathbb{N}) \) is a replacement map (or \( F \)-map) if \( \forall f \in F, \mu(f) \subseteq \{1, \ldots, \text{ar}(f)\} \) [Luc98]. Let \( MF \) be the set of all
Context-Sensitive Dependency Pairs 301

\( \mathcal{F} \)-maps (or \( M_R \) for the \( \mathcal{F} \)-maps of a TRS \( \langle \mathcal{F}, R \rangle \)). A binary relation \( R \) on terms is \( \mu \)-monotonic if \( t R s \) implies \( f(t_1, \ldots, t_{i-1}, t, t_{i+1}, \ldots, t_k) R f(t_1, \ldots, t_{i-1}, s, t_{i+1}, \ldots, t_k) \) for all \( f \in \mathcal{F}, t \in \mu(f) \), and \( t, s, t_1, \ldots, t_k \in T(\mathcal{F}, X) \). The set of \( \mu \)-replacing positions \( \text{Pos}^\mu(t) \) of \( t \in T(\mathcal{F}, X) \) is: \( \text{Pos}^\mu(t) = \{ i \} \), if \( t \notin X \) and \( \text{Pos}^\mu(t) = \{ A \} \cup \bigcup_{i \in \text{root}(t)} \mathcal{F} \cup \text{Pos}^\mu(t_i) \), if \( t \notin X \). 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 [Luc98]), we (only) contract replacing redexes: \( t \triangleright_{\mu} s \) or \( t \triangleright_{\mu} s \) (or \( t \triangleright_{\mu} s \) and even \( t \triangleright_{\mu} s \), if \( t \triangleright_{\mu} s \) and \( p \in \text{Pos}^\mu(t) \). A TRS \( R \) is \( \mu \)-terminating if \( \triangleright_{\mu} \) is terminating. A term \( t \) is \( \mu \)-terminating if there is no infinite \( \mu \)-rewrite sequence \( t = t_1 \triangleright_{\mu} t_2 \triangleright_{\mu} \cdots \triangleright_{\mu} t_{n-1} \triangleright_{\mu} t_n \cdots \) starting from \( t \). A pair \( (R, \mu) \) where \( R \) is a TRS and \( \mu \in M_R \) is often called a CS-TRS.

Dependency pairs. Given a TRS \( R = (\mathcal{F}, \mathcal{D}, \mathcal{X}) \) a new TRS \( DP(R) = (\mathcal{F}^D, \mathcal{D}(R)) \) of dependency pairs for \( R \) is given as follows: if \( f(t_1, \ldots, t_m) \rightarrow r \in R \) and \( r \in \mathcal{C}[g(s_1, \ldots, s_n)] \) for some defined symbol \( g \in \mathcal{D} \) and \( s_1, \ldots, s_n \in T(\mathcal{F}, X) \), then \( f^D(t_1, \ldots, t_m) \rightarrow g^D(s_1, \ldots, s_n) \in D(R) \), where \( f^D \) and \( g^D \) are new fresh symbols (called tuple symbols) associated to defined symbols \( f \) and \( g \) respectively [AG00]. Let \( \mathcal{D}^\mathcal{D} \) be the set of tuple symbols associated to symbols in \( \mathcal{D} \) and \( \mathcal{D}^\mathcal{D} = \mathcal{D} \cup \mathcal{D}^\mathcal{D} \). As usual, for \( t = f(t_1, \ldots, t_k) \in T(\mathcal{F}, X) \), we write \( t^D \) to denote the marked term \( f^D(t_1, \ldots, t_k) \). Conversely, given a marked term \( t = f^D(t_1, \ldots, t_k) \), where \( t_1, \ldots, t_k \in T(\mathcal{F}, X) \), we write \( t^D \) to denote the term \( f(t_1, \ldots, t_k) \in T(\mathcal{F}, X) \). Given \( T \subseteq T(\mathcal{F}, X) \), let \( T^D \) be the set \( \{ t^D \mid t \in T \} \).

A reduction pair \((\succeq, \sqsupseteq)\) consists of a stable and weakly monotonic quasi-ordering \( \succeq \), and a stable and well-founded ordering \( \sqsupseteq \) satisfying either \( \succeq \circ \succeq \subseteq \sqsupseteq \circ \sqsupseteq \) or \( \circ \succeq \subseteq \circ \sqsupseteq \). Note that monotonicity is not required for \( \sqsupseteq \).

3 Context-Sensitive Dependency Pairs

Let \( M_{\infty, \mu} \) be a set of minimal non-\( \mu \)-terminating terms in the following sense: \( t \) belongs to \( M_{\infty, \mu} \) 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 \( t \notin M_{\infty, \mu} \), then \( \text{root}(t) \) is a defined symbol. The following proposition establishes that, given a minimal non-\( \mu \)-replacing term \( t \in M_{\infty, \mu} \), 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 \( t \rightarrow r \), 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 1. Let \( R = (\mathcal{C} \cup \mathcal{D}, \mathcal{X}) \) be a TRS and \( \mu \in M_R \). Then for all \( t \in M_{\infty, \mu} \), there exist \( l \rightarrow r \in R \), a substitution \( \sigma \) and a term \( u \in M_{\infty, \mu} \), such...
that \( t \xrightarrow{A}\sigma(l) \xrightarrow{A}\sigma(r) \supseteq \mu u \) and either (1) there is a \( \mu \)-replacing subterm \( s \) of \( r \) such that \( u = \sigma(s) \), or (2) there is \( x \in \mathrm{Var}_s(r) - \mathrm{Var}_s(l) \) such that \( \sigma(x) \supseteq \mu u \).

Proposition 1 motivates the following.

**Definition 1.** Let \( R = (\mathcal{F}, R) = (\{C \cup D, R\}) \) be a TRS and \( \mu \in M_R \). We define 

\[
\mathrm{DP}(R, \mu) = \mathrm{DP}_F(R, \mu) \cup \mathrm{DP}_X(R, \mu)
\]

where:

\[
\mathrm{DP}_F(R, \mu) = \{ l \rightarrow s \mid l \rightarrow r, r \supseteq \mu s, \text{root}(s) \in D, l \not\supseteq \mu s \}
\]

and 

\[
\mathrm{DP}_X(R, \mu) = \{ l \rightarrow r \mid l \rightarrow r, l \rightarrow x \in \mathrm{DP}_X(R, \mu) \} .
\]

We extend \( \mu \in M_F \) into \( \mu^1 \in M_F \) by \( \mu^1(f) = \mu(f) \) if \( f \in \mathcal{F} \), and \( \mu^1(f) = \mu(f) \) if \( f \notin \mathcal{D} \).

A rule \( l \rightarrow r \) of a TRS \( R \) is \( \mu \)-conservative if \( \mathrm{Var}_s(r) \subseteq \mathrm{Var}_s(l) \), i.e., it does not contain migrating variables; \( R \) is \( \mu \)-conservative if all its rules are (see [Luc06]).

The following result is immediate from Definition 1.

**Proposition 2.** If \( R \) is a \( \mu \)-conservative TRS, then \( \mathrm{DP}(R, \mu) = \mathrm{DP}_F(R, \mu) \).

Therefore, in order to deal with \( \mu \)-conservative TRSs \( R \) we only need to consider the ‘classical’ dependency pairs in \( \mathrm{DP}_F(R, \mu) \).

**Example 3.** Consider the TRS \( R \):

\[
g(X) \rightarrow h(X) \quad h(d) \rightarrow g(c) \quad c \rightarrow d
\]

together with \( \mu(g) = \mu(h) = \emptyset \) [Zan97, Example 1]. \( \mathrm{DP}(R, \mu) \) is:

\[
g(X) \rightarrow h(X) \quad h(d) \rightarrow g(c)
\]

with \( \mu^1(g) = \mu^1(h) = \emptyset \).

If the TRS \( R \) 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 \( R \) [Zan97, Example 5]:

\[
\begin{align*}
\text{if}(\text{true}, X, Y) & \rightarrow X \\
\text{f}(X) & \rightarrow \text{if}(X, c, \text{f}(\text{true})) \\
\text{f} & \rightarrow \text{false}, X, Y \rightarrow Y
\end{align*}
\]

with \( \mu(\text{f}) = \{1, 2\} \). Then, \( \mathrm{DP}(R, \mu) \) is:

\[
\begin{align*}
\text{f}(X) & \rightarrow \text{if}(X, c, \text{f}(\text{true})) \\
\text{f}(\text{false}, X, Y) & \rightarrow Y
\end{align*}
\]

with \( \mu^1(\text{f}) = \{1\} \) and \( \mu(\text{if}) = \{1, 2\} \).

Now we introduce the notion of chain of CS-DPs.

**Definition 2** (Chain of CS-DPs). Let \( (R, \mu) \) be a CS-TRS. Given \( P \subseteq \mathrm{DP}(R, \mu) \), an \( (R, P, \mu^1) \)-chain is a finite or infinite sequence of pairs \( u_i \rightarrow v_i \in P \), for \( i \geq 1 \) such that there is a substitution \( \sigma \) satisfying both:

1. \( \sigma(u_i) \rightarrow_{R, \mu^1} \sigma(u_{i+1}) \), if \( u_i \rightarrow v_i \in \mathrm{DP}_F(R, \mu) \), and
2. if \( u_i \rightarrow v_i = u_j \rightarrow x_i \in \mathrm{DP}_X(R, \mu) \), then there is \( s_i \in T(F, X) \) such that \( \sigma(x_i) \supseteq \mu u_i \) and \( s_i \rightarrow_{R, \mu^1} \sigma(u_{i+1}) \).
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 $(R, P, \mu^\sharp)$-chain with $u_1 \to v_1 \in P$ as heading dependency pair is called minimal if $\sigma(u_1)^\flat \in M_{\infty, \mu}$ and all dependency pairs in $P$ occur infinitely often.

Remark 1. When an $(R, DP(R, \mu), \mu^\sharp)$-chain is written for a given substitution $\sigma$, we write $\sigma(u) \to DP(R, \mu), \mu^\sharp \sigma(v)$ for steps which use a dependency pair $u \to v \in DP(R, \mu)$ but we rather write $\sigma(u) \to DP(R, \mu), \mu^\sharp s^\flat$ for steps which use a dependency pair $u \to x \in DP_X(R, \mu)$, where $s$ is as in Definition 2.

In the following, we use $DP_1^X(R, \mu)$ to denote the subset of dependency pairs in $DP_X(R, \mu)$ whose migrating variables occur on non-$\mu$-replacing immediate subterms in the left-hand side:

$$DP_1^X(R, \mu) = \{ f^\flat(u_1, \ldots, u_k) \to x \in DP_X(R, \mu) \mid \exists i, 1 \leq i \leq k, i \notin \mu(f^\flat), x \in Var(u_i) \}$$

For instance, $DP_1^X(R, \mu) = DP_X(R, \mu)$ for the CS-TRS $(R, \mu)$ in Example 4. For this subset of CS-dependency pairs, we have the following.

Proposition 3. There is no infinite $(R, P, \mu^\sharp)$-chain with $P \subseteq DP_1^X(R, \mu)$.

The following result establishes the correctness of the context-sensitive dependency pairs approach.

Theorem 1 (Correctness). Let $R$ be a TRS and $\mu \in M_R$. If there is no infinite $(R, DP(R, \mu), \mu^\sharp)$-chain, then $R$ is $\mu$-terminating.

As an immediate consequence of Theorem 1 and Proposition 3, we have the following.

Corollary 1. Let $R$ be a TRS and $\mu \in M_R$. If $DP(R, \mu) = DP_1^X(R, \mu)$, then $R$ is $\mu$-terminating.

Example 5. Consider the following TRS $R$ [Luc98, Example 15]

$$\begin{align*}
&\text{add}(0, X) \to X \\
&\text{add}(s(X), Y) \to s(\text{add}(X, Y))
\end{align*}$$

with $\mu(\text{cons}) = \mu(\text{from}) = \emptyset$, $\mu(\text{add}) = \mu(\text{and}) = \mu(\text{if}) = \{1\}$, and $\mu(\text{first}) = \{1, 2\}$. Then, $DP(R, \mu) = DP_1^X(R, \mu)$ is:

$$\begin{align*}
&\text{ADD}(0, X) \to X \\
&\text{ADD}(s(X), Y) \to s(\text{add}(X, Y))
\end{align*}$$

Thus, by Corollary 1 we conclude the $\mu$-termination of $R$.

Now we prove that the previous CS-dependency pairs approach is not only correct but also complete for proving termination of CSR.
Theorem 2 (Completeness). Let $R$ be a TRS and $\mu \in M_R$. If $R$ is $\mu$-terminating, then there is no infinite $(R, DP(R, \mu), \mu^*)$-chain.

Corollary 2 (Characterization of $\mu$-termination). Let $R$ be a TRS and $\mu \in M_R$. $R$ is $\mu$-terminating if and only if there is no infinite $(R, DP(R, \mu), \mu^*)$-chain.

In the dependency pairs approach, the absence of infinite chains is checked by finding a reduction pair $(\geq, \sqsupseteq)$ 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 $(\geq, \sqsupseteq)$ where $\geq$ is a stable and $\mu$-monotonic quasi-ordering which is compatible with the well-founded and stable ordering $\sqsubseteq$, i.e., $\geq \circ \sqsubseteq \sqsubseteq$ or $\sqsubseteq \circ \geq \sqsubseteq$. 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 $DP_X(R, \mu)$. 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 $\sqsubseteq^\mu \sqsubseteq$) and also connect a term which is rooted by defined symbol $f$ and the corresponding dependency pair which is rooted by $f^1$ (hence the requirement $f(x_1, \ldots, x_k) \geq f^1(x_1, \ldots, x_k)$).

Theorem 3. Let $R = (F, R)$ be a TRS, $\mu \in M_F$. Then, $R$ is $\mu$-terminating if and only if there is a $\mu$-reduction pair $(\geq, \sqsupseteq)$ such that,

1. $l \geq \tau$ for all $l \rightarrow \tau \in R$,
2. $u \sqsupseteq v$ for all $u \rightarrow v \in DP_X(R, \mu)$, and
3. whenever $DP_X(R, \mu) \neq \emptyset$ we have that $\geq^\mu \sqsubseteq \geq$, where $\geq^\mu$ is the $\mu$-replacing subterm relation on $T(F, X)$, and

(a) $u \geq (\geq \cup \sqsupseteq) v$ for all $u \rightarrow v \in DP_X(R, \mu)$, $u \sqsupseteq v$ for all $u \rightarrow v \in DP_X(R, \mu)$, and $x_1, \ldots, x_k \geq f^1(x_1, \ldots, x_k)$ for all $f \in D$, or
(b) $u \geq (\geq \cup \sqsupseteq) v$ for all $u \rightarrow v \in DP_X(R, \mu)$ and $f(x_1, \ldots, x_k) \sqsupseteq f^1(x_1, \ldots, x_k)$ for all $f \in D$.

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 of) the dependency graph associated to the TRS $R$. The nodes of the dependency graph are the dependency pairs in $DP(R)$; 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 $\sigma(v) \rightarrow^* \theta(u')$.

Similarly, in the context-sensitive (CS-)dependency graph:

1. There is an arc from a dependency pair $u \rightarrow v \in DP_X(R, \mu)$ to a dependency pair $u' \rightarrow v' \in DP(R, \mu)$ if there are substitutions $\sigma$ and $\theta$ such that $\sigma(v) \rightarrow^*_{\mu, \sigma} \theta(u')$.
2. There is an arc from a dependency pair $u \rightarrow v \in DP_X(R, \mu)$ to each dependency pair $u' \rightarrow v' \in DP(R, \mu)$. 
Context-Sensitive Dependency Pairs 305

Note that the use of $\mu^\#$ (which restricts reductions on the arguments of the dependency pair symbols $f^j$) is essential: given a set of dependency pairs associated to a CS-TRS $(R, \mu)$, we have less arcs between them due to the presence of such replacement restrictions.

Example 6. Consider the CS-TRS in Example 1. $DP(R, \mu)$ is:

\[ F(a, b, X) \rightarrow F(X, X, X) \]

with $\mu^\#(F) = \{3\}$. Although the dependency graph contains a cycle (due to $\sigma(F(X, X, X)) \rightarrow^* \sigma(F(a, b, Y))$ for $\sigma(X) = \sigma(Y) = c$), the CS-dependency graph contains no cycle because it is not possible to $\mu^\#$-reduce $\theta(F(X, X, X))$ into $\theta(F(a, b, Y))$ for any substitution $\theta$ (due to $\mu^\#(F) = \{3\}$).

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 $R$ be a TRS, $\mu \in M_R$ and $\mathcal{C} \subseteq DP^1_R(R, \mu)$ be a cycle. Then, there is no minimal $(R, \mathcal{C}, \mu^\#)$-chain.

According to this and continuing Example 6, we conclude the $\mu$-termination of $R$ 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 $(R, \mu)$. Let $\text{Cap}^\mu$ be given as follows: let $D$ be a set of defined symbols (in our context, $D = D \cup D^\#$):

\[ \text{Cap}^\mu(x) = x \quad \text{if } x \text{ is a variable} \]

\[ \text{Cap}^\mu(f(t_1, \ldots, t_k)) = \begin{cases} y & \text{if } f \in D \\ \text{Cap}^\mu(f(t_1, \ldots, t_k)) & \text{otherwise} \end{cases} \]

where $y$ is intended to be a new, fresh variable which has not yet been used and given a term $s$, $[s]_j^i = \text{Cap}^\mu(s)$ if $i \in \mu(f)$ and $[s]_j^i = s$ if $i \notin \mu(f)$. Let $\text{Ren}^\mu$ be given by: $\text{Ren}^\mu(x) = y$ if $x$ is a variable and $\text{Ren}^\mu(f(t_1, \ldots, t_k)) = f([t_1]_y^i, \ldots, [t_k]_y^i)$ for every $k$-ary symbol $f$, where given a term $s \in T^4(F, X)$, $[s]_j^i = \text{Ren}^\mu(s)$ if $i \in \mu(f)$ and $[s]_j^i = s$ 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 $(R, \mu)$ 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 $v'$ are $\mu$-connectable.
Example 7. (Continuing Ex. 6) Since \(\text{RE}^\mu(\text{CAP}^\mu(\text{F}(X, X, X))) = \text{F}(X, X, Z)\) and \(\text{F}(a, b, Y)\) do not unify, we conclude (and this can be easily implemented) that the CS-dependency graph for the CS-TRS \((\mathcal{R}, \mu)\) in Example 1 has no cycle.

4.2 Checking \(\mu\)-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 \((\preceq_\varepsilon, \supseteq_\varepsilon)\) for each cycle \(\mathcal{C}\) [GA00, Theorem 11]. In our setting, we use \(\mu\)-reduction pairs.

**Theorem 4 (Use of the CS-dependency graph).** Let \(\mathcal{R} = (\mathcal{F}, \mathcal{R})\) be a TRS, \(\mu \in M_F\). Then, \(\mathcal{R}\) is \(\mu\)-terminating if and only if for each cycle \(\mathcal{C}\) in the context-sensitive dependency graph there is a \(\mu\)-reduction pair \((\preceq_\varepsilon, \supseteq_\varepsilon)\) such that \(\mathcal{R} \preceq_\varepsilon \mathcal{C} \subseteq \varepsilon \cup \supseteq_\varepsilon\), and

1. If \(\mathcal{C} \cap \text{DP}_{\varepsilon}(\mathcal{R}, \mu) = \emptyset\), then \(\mathcal{C} \cap \supseteq_\varepsilon \neq \emptyset\).
2. If \(\mathcal{C} \cap \text{DP}_{\varepsilon}(\mathcal{R}, \mu) \neq \emptyset\), then \(\varepsilon \supseteq_\varepsilon \mathcal{C}\), where \(\varepsilon_\mu\) is the \(\mu\)-replacing subterm relation on \(\mathcal{F}(\mathcal{F}, X)\).

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 \(\mathcal{C}\), we remove from \(\mathcal{C}\) those pairs \(u \rightarrow v\) satisfying \(u \supseteq v\). Then, we recompute the SCCs with the remaining pairs in the CS-dependency graph and start again (see [HM05, Section 4]). In our setting, it is not difficult to see that, if the condition \(f(x_1, \ldots, x_k) \supseteq_\varepsilon f^1(x_1, \ldots, x_k)\) for all \(f^1\) in \(\mathcal{C}\) or \(f(x_1, \ldots, x_k) \supseteq_\varepsilon f^1(x_1, \ldots, x_k)\) for all \(f^1\) in \(\mathcal{C}\).

Example 8. Consider the CS-TRS \((\mathcal{R}, \mu)\) in Example 4 and \(\text{DP}(\mathcal{R}, \mu)\):

\[
\begin{align*}
\text{F}(X) & \rightarrow \text{IF}(X, c, \text{f}(true)) \\
\text{IF}(\text{false}, X, Y) & \rightarrow Y
\end{align*}
\]

with \(\mu(\text{F}) = \{1\}\) and \(\mu(\text{IF}) = \{1, 2\}\). These two CS-dependency pairs form the only cycle in the CS-dependency graph. The \(\mu\)-reduction pair \((\supseteq_\varepsilon, >)\) induced by the polynomial interpretation

\[
\begin{align*}
[c] & = \text{true} = 0 \\
[\text{false}] & = 1 \\
\text{[f]}(x) & = x \\
\text{[if]}(x, y, z) & = x + y + z \\
\text{[F]}(x) & = x \\
\text{[IF]}(x, y, z) & = x + z
\end{align*}
\]

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 \( R \) be a TRS and \( \mathcal{C} \subseteq \text{DP}(R) \) such that every dependency pair symbol in \( \mathcal{C} \) has positive arity. A simple projection for \( \mathcal{C} \) is a mapping \( \pi \) that assigns to every \( k \)-ary dependency pair symbol \( f^k \) in \( \mathcal{C} \) an argument position \( i \in \{1, \ldots, k\} \). The mapping that assigns to every term \( f^k(t_1, \ldots, t_k) \in T^2(F, X) \) with \( f^k \) a dependency pair symbol in \( R \) its argument position \( \pi(f^k) \) is also denoted by \( \pi \).

In the following result, for a simple projection \( \pi \) and \( \mathcal{C} \subseteq \text{DP}(R, \mu) \), we let \( \pi(\mathcal{C}) = \{ \pi(u) \rightarrow \pi(v) \mid u \rightarrow v \in \mathcal{C} \} \). Note that \( u, v \in T^2(F, X) \), but \( \pi(u), \pi(v) \in T(F, X) \).

Theorem 5. Let \( R \) be a TRS and \( \mu \in M_R \). Let \( \mathcal{C} \subseteq \text{DP}_\mu(R, \mu) \) be a cycle. If there exists a simple projection \( \pi \) for \( \mathcal{C} \) such that \( \pi(\mathcal{C}) \subseteq \mathcal{C} \), and \( \pi(\mathcal{C}) \cap \mathcal{C} \neq \emptyset \), then there is no minimal \((R, \mathcal{C}, \mu^2)\)-chain.

Note that the result is restricted to cycles which do not include dependency pairs in \( \text{DP}_\mu(R, \mu) \). 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 \( R = (F, R) \) be a TRS, \( \mu \in M_F \) and \( \mathcal{C} \subseteq \text{DP}_\mu(R, \mu) \) be a cycle. Let \( \triangleright \) be a stable quasi-ordering on terms whose strict and stable part \( \triangleright \) is well-founded and \( \pi \) be a simple projection for \( \mathcal{C} \) such that for all \( f^k \) in \( \mathcal{C} \), \( \pi(f^k) \not\triangleright f^k \) and \( \pi(\mathcal{C}) \subseteq \triangleright \).

1. If \( \mathcal{C} \cap \text{DP}_\mu(R, \mu) = \emptyset \) and \( \mathcal{C} \cap \triangleright \neq \emptyset \), then there is no minimal \((R, \mathcal{C}, \mu^2)\)-chain.
2. If \( \mathcal{C} \cap \text{DP}_\mu(R, \mu) \neq \emptyset \), \( \triangleright \) is the \( \mu \)-replacing subterm relation on \( T(F, X) \), and \( \mathcal{C} \cap \triangleright \neq \emptyset \) and \( f(x_1, \ldots, x_k) \triangleright x_{\pi(f^k)} \) for all \( f \in \mathcal{D} \) such that \( f^k \) is in \( \mathcal{C} \), or \( f(x_1, \ldots, x_k) \triangleright x_{\pi(f^k)} \) for all \( f \in \mathcal{D} \) such that \( f^k \) is in \( \mathcal{C} \), then there is no minimal \((R, \mathcal{C}, \mu^2)\)-chain.

Example 9. Consider the CS-TRS \((R, \mu)\) in Example 3. \( \text{DP}(R, \mu) \) is:

\[
\begin{align*}
G(X) & \rightarrow H(X) \\
H(d) & \rightarrow G(c)
\end{align*}
\]

where \( \mu(G) = \mu(H) = \emptyset \). The dependency graph contains a single cycle including both of them. The only simple projection is \( \pi(G) = \pi(H) = 1 \). Since \( \pi(G(X)) = \pi(H(X)) = d \triangleright c = \pi(G(c)) \), we only need to guarantee that \( \pi(H(d)) = d \triangleright c = \pi(G(c)) \) holds for a stable and well-founded ordering \( \triangleright \). This is easily fulfilled by, e.g., a polynomial ordering.
6 Conclusions

We have shown how to use dependency pairs in proofs of termination of CSR. The implementation and practical use of the developed techniques yield a novel and powerful framework which improves the current state-of-the-art of methods for proving termination of CSR. Some interesting differences arise which can be summarized as follows: in sharp contrast to the standard dependency pairs approach, where all dependency pairs have tuple symbols $f^*$ 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 $\text{DP}(R, \mu)$ characterizes the $\mu$-termination of $R$ (Theorems 1 and 2). Furthermore, we are also able to use term orderings to ensure the absence of infinite chains of context-sensitive dependency pairs (Theorem 3). In fact, we are properly extending Arts and Giesl’s approach: whenever $\mu(f) = \{1, \ldots, k\}$ for all $k$-ary symbols $f \in F$, 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

Context-Sensitive Dependency Pairs 145


8.7 Improving the Context-Sensitive Dependency Graph

Improving the Context-sensitive
Dependency Graph

Beatriz Alarcón, Raúl Gutiérrez, and Salvador Lucas ¹,²

Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Valencia, Spain

Abstract
The dependency pairs method is one of the most powerful techniques 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 \( \mu : F \to \mathcal{P}(N) \) satisfying \( \mu(f) \subseteq \{1, \ldots, k\} \), for each \( k \)-ary symbol \( f \) of the signature \( F \) [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 \( f(t_1, \ldots, t_k) \) if \( i \in \mu(f) \); every term \( t \) (as a whole) is \( \mu \)-replacing by definition. For other subterms we proceed inductively.

¹ This work has been partially supported by the EU (FEDER) and the Spanish MEC, under grants TIN 2004-7943-C04-02 and IIA 2006-0007, the Generalitat Valenciana under grant GV06/285, and the ICT for EU-India Cross-Cultural Dissemination: ALA/95/23/2005/077-054 project. Beatriz Alarcón was partially supported by the Spanish MEC under FPU grant AP2005-3399.
² Email: \{balarcon,rgutierrez,slucas\}@dsic.upv.es

1571-0661/S – see front matter © 2007 Elsevier B.V. All rights reserved.
doi:10.1016/j.entcs.2007.05.040
in this way. Then, for a given TRS, we obtain a restriction of rewriting which we call context-sensitive rewriting. Proving termination of CSR is an interesting problem with several applications in the fields of term rewriting and programming languages (see [5,6,8,11,13] for further motivation).

The dependency pairs method [1] is one of the most powerful techniques for proving termination of rewriting. Roughly speaking, given a TRS $R$, the dependency pairs associated to $R$ conform a new TRS $\text{DP}(R)$ which (together with $R$) determines the so-called dependency chains whose finiteness characterizes termination of $R$. 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 $\text{mu-term}$ [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 $R$:

\[
\begin{align*}
  a & \rightarrow c(f(a)) \\
  f(c(X)) & \rightarrow X
\end{align*}
\]

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 ‘standard’ dependency pair. We could wrongly conclude that $R$ is $\mu$-terminating, which is not true:

\[
\begin{align*}
  f(a) & \rightarrow_{\mu} f(f(a)) \\
  f(c(f(a))) & \rightarrow_{\mu} f(a) \rightarrow_{\mu} \cdots
\end{align*}
\]

Indeed, as shown in [3], we must add the following dependency pair

\[
F(c(X)) \rightarrow X
\]

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 $R$. The nodes of the dependency graph are the dependency pairs in $\text{DP}(R,\mu)$. A disappointing aspect of collapsing context-sensitive dependency pairs (as $F(c(X)) \rightarrow X$ 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 outcoming links of collapsing dependency pairs to dependency pairs headed by the so-called hidden symbols which 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 $R$ which can be used to compute the list of prime numbers $\{7\}$:

<table>
<thead>
<tr>
<th>Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td>primes -&gt; sieve(from(s(0))))</td>
</tr>
<tr>
<td>from(X) -&gt; cons(X,from(s(X)))</td>
</tr>
<tr>
<td>if(true,X,Y) -&gt; X</td>
</tr>
<tr>
<td>head(cons(X,Y)) -&gt; X</td>
</tr>
<tr>
<td>if(false,X,Y) -&gt; Y</td>
</tr>
<tr>
<td>filter(s(s(X)),cons(Y,Z)) -&gt; if(divides(s(s(X)),Y),filter(s(s(X)),Z),cons(Y,filter(X,sieve(Y))))</td>
</tr>
<tr>
<td>sieve(cons(X,Y)) -&gt; cons(X,filter(X,sieve(Y)))</td>
</tr>
</tbody>
</table>

Together with $\mu(\text{cons}) = \mu(\text{if}) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for any other symbols $f$. No (automatic or manual) proof of termination for this CSTRS has been reported to date. By using the dependency graph as defined in \cite{3} we were not able to find a proof with MU-TERM 4.3 \cite{2}.

In contrast, with the new definition in this paper, we have no cycles! Thus, a direct (and automatic) proof of $\mu$-termination of $R$ 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 \cite{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.
\[ D = \{ \text{root}(l) \mid l \rightarrow r \in R \} \] and \( C = \mathcal{F} - D \).

**Context-sensitive rewriting.**

A mapping \( \mu : \mathcal{F} \rightarrow \mathcal{P}(\mathbb{N}) \) is a replacement map (or \( \mathcal{F} \)-map) if \( \forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \ldots, ar(f)\} \) [10]. Let \( M_\mathcal{F} \) be the set of all \( \mathcal{F} \)-maps (or \( M_\mathcal{R} \) for the \( \mathcal{F} \)-maps of a TRS \( (\mathcal{F}, R) \)). A binary relation \( R \) on terms is \( \mu \)-monotonic if \( t \rightarrow s \) implies \( f(t_1, \ldots, t_{i-1}, s, \ldots, t_k) \rightarrow f(t_1, \ldots, t_{i-1}, t_k) \) for every \( t, s, t_1, \ldots, t_k \in T(\mathcal{F}, \mathcal{X}) \).

The set of \( \mu \)-replacing positions \( Pos^\mu(t) \) of \( t \in T(\mathcal{F}, \mathcal{X}) \) is: \( Pos^\mu(t) = \{ 1 \} \), if \( t \in \mathcal{X} \) and \( Pos^\mu(t) = \{ 1 \} \cup \bigcup_{j \in \mu(\text{root}(t))} Pos^\mu(t_{n_j}), \) if \( t \notin \mathcal{X} \). The set of replacing variables of \( t \) is \( \text{Var}^\mu(t) = \{ x \in \text{Var}(t) \mid \exists p \in Pos^\mu(t), t_p = x \} \). The \( \mu \)-replacing subterm relation \( \geq_\mu \) is given by \( t \geq_\mu s \) if there is \( p \in Pos^\mu(t) \) such that \( s = t_p \). We write \( t \geq_\mu s \) if \( t \geq_\mu s \) and \( t \neq s \). In context-sensitive rewriting (CSR [10]), (we) (only) contract replacing redexes: \( \mu \)-rewrites to \( s \), written \( t \rightsquigarrow_\mu s \) (or \( t \rightsquigarrow_{R, \mu} s \)), if \( t \geq_\mu s \) and \( p \in Pos^\mu(t) \). A TRS \( R \) is \( \mu \)-terminating if \( \rightsquigarrow_\mu \) is terminating. A term \( t \) is \( \mu \)-terminating if there is no infinite \( \mu \)-rewrite sequence \( t = t_1 \rightsquigarrow_\mu t_2 \rightsquigarrow_\mu \cdots \rightsquigarrow_\mu t_n \rightsquigarrow_\mu \cdots \) starting from \( t \). A pair \((R, \mu)\) where \( R \) is a TRS and \( \mu \in M_\mathcal{R} \) is often called a CS-TRS.

**Dependency pairs.**

Given a TRS \( R = (\mathcal{F}, R) = (\mathcal{C} \cup \mathcal{D}, R) \) a new TRS \( DP(R) = (\mathcal{F}^2, D(R)) \) of dependency pairs for \( R \) is given as follows: if \( f(t_1, \ldots, t_m) \rightarrow r \in R \) and \( r = C[g(s_1, \ldots, s_n)] \) for some defined symbol \( g \in \mathcal{D} \) and \( s_1, \ldots, s_n \in T(\mathcal{F}, \mathcal{X}) \), then \( f^2(t_1, \ldots, t_m) \rightarrow g^2(s_1, \ldots, s_n) \in D(R) \), where \( f^2 \) and \( g^2 \) are new fresh symbols (called tuple symbols) associated to defined symbols \( f \) and \( g \) respectively [1]. Let \( \mathcal{D}^f \) be the set of tuple symbols associated to symbols in \( \mathcal{D} \) and \( \mathcal{F}^2 = \mathcal{F} \cup \mathcal{D}^f \). As usual, for \( t = f(t_1, \ldots, t_k) \in T(\mathcal{F}, \mathcal{X}) \), we write \( t^2 \) to denote the marked term \( f^2(t_1, \ldots, t_k) \). Conversely, given a marked term \( t = f^2(t_1, \ldots, t_k) \), where \( t_1, \ldots, t_k \in T(\mathcal{F}, \mathcal{X}) \), we write \( t^2 \) to denote the term \( f(t_1, \ldots, t_k) \in T(\mathcal{F}, \mathcal{X}) \). Given \( T \subseteq T(\mathcal{F}, \mathcal{X}) \), let \( T^2 \) be the set \{ \( t^2 \mid t \in T \) \}.

**3 Structure of infinite \( \mu \)-rewrite sequences**

Let \( M_{\infty, \mu} \) be a set of minimal non-\( \mu \)-terminating terms in the following sense: \( t \) belongs to \( M_{\infty, \mu} \) if \( t \) is non-\( \mu \)-terminating and every strict \( \mu \)-replacing subterm \( s \) of \( t \) (i.e., \( t \geq_\mu s \)) is \( \mu \)-terminating. Obviously, if \( t \in M_{\infty, \mu} \), then root \( (t) \) is a defined symbol. Furthermore, since \( \mu \)-terminating terms are preserved under \( \mu \)-rewriting, it follows that \( M_{\infty, \mu} \) is also preserved under inner \( \mu \)-rewritings.

**Lemma 3.1** Let \( R \) be a TRS and \( \mu \in M_R \). Let \( t \in M_{\infty, \mu} \). If \( t \rightsquigarrow_\mu^* s \), then \( s \in M_{\infty, \mu} \).

The following proposition establishes that, given \( t \in M_{\infty, \mu} \), 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 \(R = (C \uplus D, R)\) be a TRS and \(\mu \in M_R\). For all \(t \in M_{\infty,\mu}\), there exist \(l \rightarrow r \in R\), a substitution \(\sigma\) and a term \(u \in M_{\infty,\mu}\), such that \(t \xrightarrow{\sigma} \sigma(l) \xrightarrow{\sigma} \sigma(r) \supseteq\mu u\) 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 \(\sigma(x) \supseteq\mu u\).

Now we investigate the structure of such sequences in more detail. In the following, we write \(t \triangleright_p s\) to denote that \(s\) is a non-replacing (hence strict!) subterm of \(t\): \(t \triangleright_p s\) if there is \(p \in \text{Pos}(t) - \text{Pos}^\mu(t)\) such that \(s = t_p\).

**Definition 3.3** [Hidden symbol] Let \(R = (F, R)\) be a TRS and \(\mu \in M_R\). We say that \(f \in F\) is a hidden symbol if there is a rule \(l \rightarrow r \in R\) and \(t \in T(F, X)\) such that \(r \triangleright f t\) and \(\text{root}(t) = f\). Let \(H(R, \mu)\) (or just \(H\), if \(R\) and \(\mu\) are clear for the context) be the set of all hidden symbols in \((R, \mu)\).

**Lemma 3.4** Let \(R = (F, R)\) be a TRS and \(\mu \in M_R\). Let \(t \in T(F, X)\) and \(\sigma\) be a substitution. If there is a rule \(l \rightarrow r \in R\) such that \(\sigma(l) \triangleright t\) and \(\text{root}(t) = \sigma(l)\), then there is no \(x \in \text{Var}(r)\) such that \(\sigma(x) \triangleright t\). Furthermore, there is a term \(t' \in T(F, X)\) such that \(r \triangleright t'\), \(\sigma(t') = t\) 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 \(\sigma(r) \triangleright t\), then there is a non-variable and non-replacing position \(p \in \text{Pos}_X(r) - \text{Pos}^\mu(r)\), such that \(\text{root}(r_p) = \text{root}(t) \in H(R, \mu)\) and \(\sigma(r_p) = t\). Then, we let \(t' = r_p\).

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 \(R = (F, R)\) be a TRS and \(\mu \in M_R\). Let \(A\) be a finite \(\mu\)-rewrite sequence \(t_1 \leftarrow t_2 \leftarrow \cdots \leftarrow t_n\) with \(t_i \in M_{\infty,\mu}\) for all \(i, 1 \leq i \leq n\) and \(n \geq 1\). If there is a term \(t \in M_{\infty,\mu}\) such that \(t_1 \triangleright t\) and \(t_n \triangleright t\), 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 \(t_i \triangleright t\) and \(t_n \triangleright t\). Let \(l \rightarrow r \in R\) be such that \(t_{n-1} = C[\sigma(l)]\) and \(t_n = C[\sigma(r)]\) for some context \(C[\cdot]\). We consider two cases: either \(t_{n-1} \triangleright t\) holds or not.

(a) If \(t_{n-1} \triangleright t\), then by the induction hypothesis we have that \(\text{root}(t) \in \mathcal{H}\).

(b) If \(t_{n-1} \triangleright t\) does not hold, then one of the following cases holds:

1. \(t_{n-1} \triangleright t\); then \(t_{n-1} \in M_{\infty,\mu}\) implies that \(t \notin M_{\infty,\mu}\), leading to a contradiction.
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 \( \mathcal{R} = (\mathcal{F}, \mathcal{R}) \) be a TRS and \( \mu \in M_{\mathcal{R}} \). Let \( A \) be an infinite sequence of the form \( t_1 \xleftarrow{s_1} t_2 \xrightarrow{\mu} t_3 \xleftarrow{s_2} t_4 \xrightarrow{\mu} t_5 \xleftarrow{s_3} t_6 \xrightarrow{\mu} \cdots \) with \( t_i, t'_i \in M_{\infty, \mu} \) for all \( i \geq 1 \). If there is a term \( t \in M_{\infty, \mu} \) such that \( t_i \triangleright_{\mu} t \) for some \( i \geq 1 \), then \( \text{root}(t) \in H \cap D \) 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 \( t_{i-1} \triangleright_{\mu} t \) does not hold, then let \( l \rightarrow r \in \mathcal{R} \) 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 \( t_{i-1} \in M_{\infty, \mu} \) it would imply that \( t \notin M_{\infty, \mu} \), 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 \( t'_i, t \in M_{\infty, \mu} \) 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 \( \text{root}(t) \in H \). Since \( t \in M_{\infty, \mu} \), 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 \( t'_i \triangleright t \) we conclude \( \text{root}(t) \in \mathcal{H} \cap \mathcal{D} \).

As an immediate consequence of Proposition 3.6, we have the following result which we will use later.

**Corollary 3.7** Let \( (\mathcal{R}, \mu) \) be a CS-TRS, \( A \) be an infinite sequence of the form \( t_1 \xleftarrow{s_1} t_2 \xrightarrow{\mu} t_3 \xleftarrow{s_2} t_4 \xrightarrow{\mu} t_5 \xleftarrow{s_3} t_6 \xrightarrow{\mu} \cdots \) with \( t_i, t'_i \in M_{\infty, \mu} \) for all \( i \geq 1 \). If there is a term \( t \in M_{\infty, \mu} \) such that \( t_i \triangleright_{\mu} t \) for some \( i \geq 1 \) and \( \text{root}(t) \in \mathcal{D} \), 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 \( (\mathcal{R}, \mu) = (\mathcal{C} \cup \mathcal{D}, \mathcal{R}) \) be a TRS and \( \mu \in M_{\mathcal{R}} \). We define \( \text{DP}(\mathcal{R}, \mu) = \text{DP}_{\mathcal{C}}(\mathcal{R}, \mu) \cup \text{DP}_{\mathcal{D}}(\mathcal{R}, \mu) \) to be the set of
context-sensitive dependency pairs (CS-DPs) where:

$$\text{DP}_\mathcal{F}(\mathcal{R}, \mu) = \{ l \rightarrow s^\sharp | l \rightarrow r \in \mathcal{R}, r \geq_{\mu} s, \text{root}(s) \in \mathcal{D}, l \not\mu s \}$$

and $$\text{DP}_\mathcal{X}(\mathcal{R}, \mu) = \{ l \rightarrow x | l \rightarrow r \in \mathcal{R}, x \in \text{Var}^\mu(r) - \text{Var}^\mu(l) \}$$.

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 $$\rightarrow$$ SIEVE($$\text{from}(s(s(0)))$$)
2. PRIMES $$\rightarrow$$ FROM($$s(q(0)))$$
3. TAIL($$\text{cons}(X,Y)) \rightarrow Y$$
4. IF(true,X,Y) $$\rightarrow$$ X
5. IF(false,X,Y) $$\rightarrow$$ Y
6. FILTER($$s(X), \text{cons}(Y, Z)) \rightarrow$$
   IF(divides($$s(s(X)), Y), \text{filter}(s(s(X)), Z), \text{cons}(Y, \text{filter}(X, \text{ sievel}(Y))))$$

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) $$\sigma(u_i) \leftarrow_{\mu^\sharp}^* \sigma(v_{i+1})$$, 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 $$\sigma(x_i) \geq_{\mu} s_i$$ and $$s_i \leftarrow_{\mu^\sharp}^* \sigma(u_{i+1})$$.

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 $$i \geq 1$$ $$\sigma(u_i) \in M_{\infty, \mu}$$, $$s_i \in M_{\infty, \mu}$$ (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 $$\sigma(u) \leftarrow_{\text{DP}(\mathcal{R}, \mu), \mu^\sharp} \sigma(v)$$ for steps which use a dependency pair $$u \rightarrow v \in \text{DP}_\mathcal{F}(\mathcal{R}, \mu)$$ but we rather write $$\sigma(u) \leftarrow_{\text{DP}(\mathcal{R}, \mu), \mu^\sharp} s^\sharp$$ 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 $$\mathcal{C}$$ 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 $$\sigma(v) \leftarrow_{\mathcal{R}, \mu^\sharp} \sigma(u')$$.

(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 \((\mathcal{R},\mathcal{P},\mu^t)\)-chain involving an infinite number of dependency pairs \(u_i \rightarrow v_i \in \text{DP}_X(\mathcal{R},\mu)\) such that \(\text{root}(u_{i+1})^2 \notin \mathcal{H}\).

**Proof.** By contradiction. Let \(A\) be an infinite \((\mathcal{R},\mathcal{P},\mu^t)\)-minimal chain of CS-DPs characterized by the CS-DPs \(u_i \rightarrow v_i\) for \(i \geq 1\):

\[
\sigma(u_1) \xrightarrow{\mathcal{C}} \sigma(u_2) \xrightarrow{\mathcal{C}} \cdots
\]

where, \(s_i^t = \sigma(v_i)\) if \(u_i \rightarrow v_i \in \text{DP}_X(\mathcal{R},\mu)\) and \(\sigma(x_i) \geq_{\mu} s_i\) for all \(i \in I\), \(u_i \rightarrow v_i \in \text{DP}_X(\mathcal{R},\mu)\) and \(\text{root}(u_{i+1})^2 \notin \mathcal{H}\). Given \(\eta(i)\) be the ‘next’ positive integer in \(I\): \(\eta(i) = \min\{j \in I \mid j > i\}\). Obviously, for all \(i \in I\), \(\eta(i) \in I\).

As a consequence of this result, we can dismiss the arcs of the dependency graph which connect collapsing dependency pairs \(u_i \rightarrow v_i\) that originate the dependency pairs \(u_i \rightarrow v_i\) which are used in \(A\):

\[
\sigma(u_1)^2 \xrightarrow{\mathcal{C}} \sigma(v_1) \xrightarrow{\mathcal{C}} u_2 \xrightarrow{\mathcal{C}} \sigma(v_2) \xrightarrow{\mathcal{C}} \cdots
\]

which corresponds to \(A\) above: by minimality of \(A\) (see Definition 4.3), we have that \(\sigma(v_i)^2\) is a positive integer in \(\mathcal{M}_{\infty,\mu}\) for all \(i \geq 1\). By definition of \(I\), for all \(i \in I\), \(v_i = x_i \in X\). By definition of collapsing dependency pair, \(x_i \in \text{Pos}(u_i)\) \(\Rightarrow \text{Pos}^\mu(u_i)\) and \(\sigma(x_i) \geq_{\mu} s_i\). Thus, \(\sigma(u_i) \geq_{\mu} s_i\) for all \(i \in I\). By repeatedly applying Corollary 3.7, we have that \(\sigma(u_i) \geq_{\mu} \sigma(u_{\eta(i)})\), i.e., \(\sigma(u_i) \geq_{\mu} \sigma(u_{\eta(i)})\) for all \(i \in I\). Thus, we obtain an infinite \(\geq_{\mu}\)-sequence which contradicts well-foundedness of \(\geq_{\mu}\). \(\square\)

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 \(\text{root}(u')^2 \notin \mathcal{H}\). 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 \(u \rightarrow v \in \text{DP}_F(\mathcal{R},\mu)\) to a dependency pair \(u' \rightarrow v' \in \text{DP}(\mathcal{R},\mu)\) if there is a substitution \(\sigma\) such that \(\sigma(v) \leq^\sigma_{\mathcal{R},\mu} \sigma(u')\).

(ii) There is an arc from a dependency pair \(u \rightarrow v \in \text{DP}_X(\mathcal{R},\mu)\) to a dependency pair \(u' \rightarrow v' \in \text{DP}(\mathcal{R},\mu)\) if \(\text{root}(u')^2 \in \mathcal{H}(\mathcal{R},\mu)\).

**Example 4.8** Consider again the TRS \(\mathcal{R}\) in Example 1.2. The hidden defined symbols are \texttt{filter}, \texttt{from}, and \texttt{sieve}. The dependency graph which corresponds to this example is shown in Figure 1 (right). Note that, in contrast to the situation...
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]

\[
\begin{align*}
  f(X) & \rightarrow \text{cons}(X,f(g(X))) \\
  g(0) & \rightarrow s(0) \\
  g(s(X)) & \rightarrow s(s(g(X))) \\
  \text{sel}(0,\text{cons}(X,Y)) & \rightarrow X \\
  \text{sel}(s(X),\text{cons}(Y,Z)) & \rightarrow \text{sel}(X,Z)
\end{align*}
\]

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:

\[
\begin{align*}
  G(s(X)) & \rightarrow G(X) \\
  \text{SEL}(s(X),\text{cons}(Y,Z)) & \rightarrow \text{SEL}(X,Z) \\
  \text{SEL}(s(X),\text{cons}(Y,Z)) & \rightarrow Z
\end{align*}
\]

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]

\[
\begin{align*}
f(0) & \rightarrow \text{cons}(0,f(s(0))) \\
f(s(0)) & \rightarrow f(p(s(0))) \\
p(s(X)) & \rightarrow X
\end{align*}
\]

together with \( \mu(f) = \mu(p) = \mu(s) = \mu(\text{cons}) = \{1\} \) and \( \mu(0) = \emptyset \). Then \( \text{DP}(R,\mu) \) is:

\[
\begin{align*}
F(s(0)) & \rightarrow F(p(s(0))) \\
F(s(0)) & \rightarrow F(s(0))
\end{align*}
\]

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 \( R \) 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 \( (R,\mu) \) be a CSTRS. A term \( t \) \( \mu \)-narrows to a term \( s \) (written \( t \sim_{\mu} s \)), 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 \( R \) (sharing no variable with \( t \)), and \( s = \theta(t[r]^p) \).

To achieve more precision when connecting two CS-DPs in a \( (R,\text{DP}(R,\mu),\mu^2) \)-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 \( v_1, \ldots, v_n \) which are \( \mu \)-narrowings of \( v \) with unifier \( \theta_i \) for \( i \in \{1,\ldots,n\} \) and can be used instead of \( v \). Not only the right-hand sides of the CS-dependency pairs are \( \mu \)-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 \( \theta_1(u) \rightarrow v_1, \ldots, \theta_n(u) \rightarrow v_n \).

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 \( R \) be a TRS and let \( P \) be a set of CS-dependency pairs. Let \( u \rightarrow v \in P \) such that \( v \) is linear and for all \( u' \rightarrow v' \in P \) (with renamed variables) the terms \( v \) and \( u' \) are not unifiable. Let
\[
P' = (P - \{u \rightarrow v\}) \cup \{u' \rightarrow v' \mid u' \rightarrow v' \text{ is a narrowing of } u \rightarrow v\}.
\]
There exists an infinite \((R, P, \mu)\)-chain iff there exists an infinite \((R, P', \mu')\)-chain.

Proof. The proof of this theorem corresponds to the proof of Theorem 25 in [1]. Note that only dependency pairs in \( DP(R, \mu) \) 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 \( P \). 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.

Thus, after narrowing the dependency pairs in \( DP(R, \mu) \) 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 \( \mu(f) = \{1\} \)) 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 \textsc{mu-term} [2,12]. We have used our new implementation to compare with the last version of the tool: \textsc{mu-term} 4.3. The benchmarks were executed in a completely automatic way (see [2] for a description of \textsc{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:

\texttt{http://www.lri.fr/~marche/termination-competition/2006}

As remarked above, our termination expert works as explained in [2] for version 4.3 of \textsc{mu-term}. For the new version 4.4 of \textsc{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 \textsc{mu-term} (corresponding to [3]).
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 $R$ as a proof of termination of a transformed TRS $R_{\Theta}^{\mu}$ (where $\Theta$ represents the transformation). If we are able to prove termination of $R_{\Theta}^{\mu}$ (using the standard methods), then the $\mu$-termination of $R$ 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 outcoming links of collapsing dependency pairs to dependency pairs headed by the so-called hidden symbols. Hidden symbols are defined symbols that occur in non-replacing positions in the right-hand sides of some rule in the TRS. This greatly improves the performance of termination proofs based on the dependency graph proposed in [3]. Narrowing context-sensitive dependency pairs has also been investigated. It can also be helpful to simplify or restructure the dependency graph and eventually simplify the proof of termination. Regarding the practical use of the (refinements on the) new CS-dependency graph in proofs of termination of CSR, we have implemented these ideas as part of the termination tool MU-TERM and we have obtained quite good results in terms of new examples which could be proved, and also regarding the time for achieving the proofs.

Since the state-of-the-art of DP-based techniques for proving termination of CSR which has been introduced in this paper corresponds to the 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.

Table 1

<table>
<thead>
<tr>
<th>Termination Tool</th>
<th>Total</th>
<th>CS-DPs</th>
<th>CSRPO</th>
<th>Transf.</th>
<th>Average time</th>
</tr>
</thead>
<tbody>
<tr>
<td>MU-TERM (PROLE'06)</td>
<td>66</td>
<td>65</td>
<td>0</td>
<td>1</td>
<td>1.68s</td>
</tr>
<tr>
<td>MU-TERM (FST&amp;TCS'06)</td>
<td>56</td>
<td>45</td>
<td>7</td>
<td>4</td>
<td>4.55s</td>
</tr>
<tr>
<td>AProVE</td>
<td>56</td>
<td>0</td>
<td>0</td>
<td>56</td>
<td>4.74s</td>
</tr>
</tbody>
</table>
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


8.8 Proving Termination of Context-Sensitive Rewriting with MU-TERM

Proving Termination of Context-Sensitive Rewriting with MU-TERM

Beatriz Alarcón, Raúl Gutiérrez, José Iborra and Salvador Lucas

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
we call context-sensitive rewriting. A TRS $R$ together with a replacement map $\mu$ is often called a CS-TRS and written $(R,\mu)$.

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 $(R,\mu)$:

- **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 $R^\mu_\Theta$ (where $\Theta$ represents the transformation). If we are able to prove termination of $R^\mu_\Theta$ (using the standard methods), then termination of the CS-TRS is ensured (see [22] for a recent survey).

MU-TERM was the first tool implementing techniques for proving termination of CSR [19]. The tool is available here:

http://www.dsic.upv.es/~slucas/csr/termination/muterm

Nowadays, the tool AProVE [15] also accepts context-sensitive termination problems specified in the TPDB format\(^3\). However, AProVE’s proofs of termination of CSR are based on using transformations (i.e., no direct proof method is currently available). The new version of MU-TERM which we present here implements all currently known techniques. The new contributions which we report in this paper are the following:

(i) We have implemented the context-sensitive recursive path ordering described in [4].

(ii) We have implemented the context-sensitive dependency pairs approach described in [2].

(iii) On the basis of recent theoretical and experimental results (see [22]), we have developed a termination expert for CSR which combines the different existing techniques for proving termination of CSR without any interaction with the user.

Finally, we want to mention that the Maude Termination Tool [9]:

http://www.lcc.uma.es/~duran/MTT

which transforms proofs of termination of Maude programs into proofs of termination of CSR uses MU-TERM’s expert as an auxiliary tool.

We assume a basic knowledge about term rewriting, see [24] for missing definitions and more information. In Section 2 we briefly describe the new features which have been added to MU-TERM. Section 3 discusses the termination expert. Section 4 provides an experimental evaluation of the new version of MU-TERM. Section 5 concludes and discusses future work.

\(^3\) See http://www.lri.fr/~marche/tpdb/format.html
MU-TERM is written in Haskell, and wxHaskell 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.

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]).
In the following, we briefly describe the new features implemented in the current version of MU-TERM.

2.1 Context-Sensitive Recursive Path Ordering (CSRPO)

CSRPO extends the recursive path ordering (RPO [7]) to context-sensitive terms [4]. The first idea that comes in mind to extend RPO to CSR (CSRPO) is marking the symbols which are in blocked positions and consider them smaller than the active ones. Therefore, terms in blocked positions become smaller. However, marking all symbols in non-replacing positions can unnecessarily weaken the resulting ordering. Thus, in addition to the usual precedence\(^6\) \(\succeq\) on the symbols of the signature \(\mathcal{F}\) of the TRS, a marking map, denoted by \(m\), is also used. The marking map defines, for every symbol and every blocked position, the set of symbols that should be marked. By \(\mathcal{F}\) we denote the set of marked symbols corresponding to \(\mathcal{F}\). Given a \(k\)-ary symbol \(f\) in \(\mathcal{F} \cup \mathcal{F}\) and \(i \in \{1, \ldots, k\}\), a marking map \(m\) provides the subset of symbols in \(\mathcal{F}\) that should be marked, i.e., \(m(f,i) \subseteq \mathcal{F}\). Marking maps are intended to mark only blocked arguments, i.e., \(m(f,i) = \emptyset\) 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 \(\mathcal{R}\):

\[
\begin{align*}
\text{from}(X) & \rightarrow \text{cons}(X,\text{from}(s(X))) \\
\text{sel}(0,\text{cons}(X,Y)) & \rightarrow X \\
\text{sel}(s(X),\text{cons}(Y,Z)) & \rightarrow \text{sel}(X,Z)
\end{align*}
\]

together with \(\mu(\text{cons}) = \mu(s) = \mu(\text{from}) = \{1\}\) and \(\mu(\text{sel}) = \{1, 2\}\). 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):

\[
\text{sel} \succeq \text{from} \succeq \text{cons} \succeq s
\]

\(m(\text{cons}, 2) = m(\text{cons}, 2) = \{\text{from}\},\ m(\text{from}, 1) = \emptyset\)

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:

\[
\begin{align*}
[\text{from}] (X) &= 3X + 2 \\
[\text{cons}] (X,Y) &= X + \frac{1}{4}Y \\
[\text{s}] (X) &= 2X + 1 \\
[\text{sel}] (X,Y) &= 2X^2Y + X + Y + 1
\end{align*}
\]

the proof using CSRPO is much faster.

2.2 Context-Sensitive Dependency Pairs (CSDPs)

Recently, the dependency pairs approach [1], one of the most powerful techniques for proving termination of rewriting, has been generalized to be used in proofs of

---

\(^6\) By a precedence, we mean a reflexive and transitive relation.
Roughly speaking, given a TRS $R$, the dependency pairs $u \rightarrow v$ associated to $R$ conform a new TRS $DP(R)$ which (together with $R$) determines the so-called dependency chains whose finiteness or infiniteness characterize termination of $R$. 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 $R$) from $\sigma(v)$ to $\sigma(v')$. These ideas are generalized (with a number of non-trivial changes) to CSR.

**Example 2.2** Consider the following non-terminating TRS $R$ borrowing the well-known Toyama’s example [12, Example 1]:

$$
\begin{align*}
&f(a, b, X) \rightarrow f(X, X, X) \\
&c \rightarrow a \\
&c \rightarrow b
\end{align*}
$$

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 $R$!) 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 $R$ 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
proving termination of CSR, only the complete Giesl and Middeldorp’s transformation \([13]\) (yielding a TRS \(R^C_\mu\)) could be used in this case, but no concrete proof of termination for \(R^C_\mu\) is known yet. Furthermore, \(R^\mu\) 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 \(R\) 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 \(R\) \([26, \text{Example 5}]\): 

\[
\begin{align*}
\text{if}(\text{true}, X, Y) & \rightarrow X \\
\text{if}(\text{false}, X, Y) & \rightarrow Y \\
\text{f}(X) & \rightarrow \text{if}(X, c, \text{f}(\text{true})) \\
\text{IF}(X, c, \text{f}(\text{true})) & \rightarrow \text{Y}
\end{align*}
\]

with \(\mu(\text{if}) = \{1, 2\}\). There are two dependency pairs:

\[
\begin{align*}
\text{f}(X) & \rightarrow \text{IF}(X, c, \text{f}(\text{true})) \\
\text{IF}(\text{false}, X, Y) & \rightarrow \text{Y}
\end{align*}
\]

with \(\mu\) extended by \(\mu(\text{f}) = \{1\}\) and \(\mu(\text{IF}) = \{1, 2\}\).

A direct and automatic proof of \(\mu\)-termination of \(R\) 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 \(R\) in Example 2.3 is not possible by using CSRPO. Furthermore, the \(\mu\)-termination of \(R\) cannot be proved by using a polynomial or-
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) \( \{0,1,2\} \), 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.
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 \( R = (F, R) \), we use parametric polynomial interpretations for the symbols \( f \in F \), 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 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, \( LL(R) \) means that \( R \) is left-linear, \( CM_R \) is the set of replacement maps which are not more restrictive than the canonical replacement map \( \mu^\text{can}_R \) of the TRS \( R \). 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, \( SN(R) \) represents a check of termination of the TRS \( R \). 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:


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


8.9 Usable Rules for Context-Sensitive Rewrite Systems

Usable Rules for Context-Sensitive Rewrite Systems

Raúl Gutiérrez¹, Salvador Lucas¹, and Xavier Urbain²

¹ DSIC, Universidad Politécnica de Valencia, Spain
{rgutierrez, slucas}@dsic.upv.es
² 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) \( R \), the dependency pairs associated to \( R \) give rise to a new TRS DP(\( R \)) which (together with \( R \)) determines the so-called dependency chains whose finiteness characterizes termination of \( R \). 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 \( C \subseteq \text{DP}(R) \) in the dependency graph, we require \( l \succeq r \) for all rules in the TRS \( R \), \( u \succeq v \) or \( u \sqcup v \) for all dependency pairs \( u \rightarrow v \in C \) and \( u \sqcup v \) for at least one \( u \rightarrow v \in C \). Here, \( \succeq \) is a stable

\footnote{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 \( U(R, C) \subseteq R \) are associated to a given cycle \( C \) of the dependency graph for \( R \). For particular (but widely used) classes of quasi-orderings \( \preceq \), we can restrict the comparisons \( l \preceq r \) to rules \( l \rightarrow r \) in \( U(R, C) \) instead of using \( R \). Since \( U(R, C) \) is (usually) smaller than \( R \), 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 \( \mu : F \rightarrow \wp(N) \) satisfying \( \mu(f) \subseteq \{1, \ldots, k\} \), for each \( k \)-ary symbol \( f \) of a signature \( F \)) 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 \( R \) 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:

\[
\begin{align*}
\text{sel}(0, \text{cons}(x, xs)) & \rightarrow x & (1) & \text{sel}(\text{cons}(n, x), \text{cons}(x, xs)) & \rightarrow \text{sel}(n, xs) & (7) \\
\text{minus}(x, 0) & \rightarrow x & (2) & \text{minus}(\text{cons}(x, y)) & \rightarrow \text{minus}(x, y) & (8) \\
\text{quot}(0, n(y)) & \rightarrow 0 & (3) & \text{quot}(\text{cons}(x, y)) & \rightarrow \text{quot}(\text{cons}(\text{minus}(x, y, n(y)))) & (9) \\
\text{zWquot}(\text{nil}, x) & \rightarrow \text{nil} & (4) & \text{from}(x) & \rightarrow \text{cons}(\text{from}(x)) & (10) \\
\text{zWquot}(\text{cons}(x, xs), \text{cons}(y, ys)) & \rightarrow \text{cons}(\text{quot}(x, y), \text{zWquot}(xs, ys)) & (12) \\
\text{head}(\text{cons}(x, xs)) & \rightarrow x & (6) \\
\text{tail}(\text{cons}(x, xs)) & \rightarrow xs & (11) \\
\end{align*}
\]
with $\mu(\text{cons}) = \{1\}$ and $\mu(f) = \{1, \ldots, \text{ar}(f)\}$ for all other symbols $f \in \mathcal{F}$. The set of CS-DPs of $\mathcal{R}$ is:

$$\begin{align*}
\text{MINUS}(s(x), s(y)) &\rightarrow \text{MINUS}(x, y) & \text{SEL}(s(n), \text{cons}(x, xs)) &\rightarrow \text{SEL}(n, xs) \\
\text{QUOT}(s(x), s(y)) &\rightarrow \text{MINUS}(x, y) & \text{ZQUOT}(\text{cons}(x, xs), \text{cons}(y, ys)) &\rightarrow \text{QUOT}(x, y) \\
\text{QUOT}(s(x), s(y)) &\rightarrow \text{QUOT}(\text{minus}(x, y), s(y)) & \text{SEL}(s(n), \text{cons}(x, xs)) &\rightarrow xs \\
\text{TAIL}(\text{cons}(x, xs)) &\rightarrow xs
\end{align*}$$

Note that non-$\mu$-replacing subterms in right-hand sides (e.g., $\text{from}(s(x))$ in rule (10)) are not considered to build the CS-DPs. Also, in sharp contrast with the unrestricted case, collapsing dependency pairs do not occur.

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 $\mathcal{F} = \mathcal{C} \cup \mathcal{D}$ 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 $\mu : \mathcal{F} \rightarrow \mathbb{N}$ is a replacement map (or $\mathcal{F}$-map) if $\forall f \in \mathcal{F}, \mu(f) \subseteq \{1, \ldots, \text{ar}(f))\}$ [18]. Let $M_\mathcal{F}$ be the set of all $\mathcal{F}$-maps ($M_\mathcal{F}$ for the $\mathcal{F}$-maps of a TRS $\mathcal{R} = (\mathcal{F}, R)$). A binary relation $R$ on terms in $T(\mathcal{F}, X)$ is $\mu$-monotonic if $t Rs$ implies $f(t_1, \ldots, t_\mu, s, \ldots, t_n) R f(t_1, \ldots, t_\mu, t, \ldots, t_n)$ for all $f \in \mathcal{F}$, $i \in \mu(f)$, and $t, s, t_1, \ldots, t_n \in T(\mathcal{F}, X)$. The set of $\mu$-replacing positions $\text{Pos}_\mu(t)$ of $t \in T(\mathcal{F}, X)$ is $\text{Pos}_\mu(t) = \{i\}$, if $t \in X$ and $\text{Pos}_\mu(t) = \{i\} \cup \bigcup_{i \in \mu(\text{root}(t))} i\text{Pos}_\mu(t|_i)$, if $t \notin X$. The set of $\mu$-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 defined 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 $t \neq s$. We write
Context-Sensitive Dependency Pairs. Given a TRS \( \mathcal{R} = (F, R) \) and \( \mu \in M_F \), the set of context-sensitive dependency pairs (CS-DPs) is \( DP(\mathcal{R}, \mu) = DP_F(\mathcal{R}, \mu) \cup DP_X(\mathcal{R}, \mu) \), where \( DP_F(\mathcal{R}, \mu) \) and \( DP_X(\mathcal{R}, \mu) \) are obtained as follows: let \( F(t_1, \ldots, t_m) \rightarrow r \in R \) and \( s \in T(F, X) \) such that \( r \triangleright s \). Then

1. if \( s = g(s_1, \ldots, s_n) \) for some \( g \in \Delta, s_1, \ldots, s_n \in T(F, X) \) and \( l \triangleright s \), then \( f^1(t_1, \ldots, t_m) \rightarrow g^1(s_1, \ldots, s_n) \in DP_F(\mathcal{R}, \mu) \); or

2. if \( s = x \in \text{Var}^\mu(r) \cap \text{Var}^\mu(l) \) and \( f^1(t_1, \ldots, t_m) \rightarrow x \in DP_X(\mathcal{R}, \mu) \). Here, \( f^1 \) and \( g^1 \) are new fresh symbols (called 

Usage Rules for Context-Sensitive Rewrite Systems 129
v′ ∈ DP(R, μ) if there is a substitution σ such that σ(v) ↪→∗R,μ σ(u′); and, there is an arc from u → v ∈ DP,X(R, μ) to u′ → v′ ∈ DP(R, μ) if root(u′) ∈ H. We consider the strongly connected components in this graph. A μ-reduction pair (⪰, =) consists of a stable and weakly μ-monotonic quasi-ordering ⪰, and a stable and well-founded ordering = satisfying ⪰ ◦ = ⊆ = or = ◦ ⪰ ⊆ =. From now on, we assume that all CS-TRSs are finite.

3 Basic Usable Rules

Consider a set of pairs P and a CS-TRS (R, μ). Then, the set of usable rules is the smallest set of rules from R which are needed to capture all the infinite minimal (R, P, μ♯)-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 R = (F, R), we say that f ∈ F directly depends on g ∈ F, written f ⪰d g, if there is a rule l → r ∈ R with f = root(l) and g occurs in r.

The set of defined symbols in a term t is DFun(t) = {f | ∃p ∈ Pos(t), f = root(t|p) ∈ D}. Let ⪰d be the transitive and reflexive closure of ⪰d. Then, we have:

**Definition 2 (Usable Rules [14,17]).** For a set G of symbols we denote by R|G the set of rewriting rules l → r ∈ R with root(l) ∈ G. The set U(R, t) of usable rules of a term t is defined as U(R, t) = ⋃l→r∈U(R,l)U(R,r). The set U(R, P) can be used instead of R when looking for a reduction pair that proves termination of R [14,17].

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 → r ∈ R is conservative if Varμ(r) ⊆ Varμ(l). First, we adapt Definition 1 to the CSR setting as follows:

**Definition 3 (Basic μ-Dependency).** Given a CS-TRS (F, μ, R), we say that f ∈ F has a basic μ-dependency on g ∈ F, written f ⪰dμ g, if there is l → r ∈ R with f = root(l) and g occurs in r at a μ-replacing position.

This leads to a straightforward extension of Definition 2. The set of μ-replacing defined symbols in a term t is DFunμ(t) = {f | ∃p ∈ Posμ(t), f = root(t|p) ∈ D}. Then, we have:

1 Note that, due to the focus on innermost CSR, [4, Def. 5] slightly differs from ours.
Definition 4 (Basic Context-Sensitive Usable Rules). Let $R = (F, R)$ be a TRS and $\mu \in M_R$. The set $U_B(R, \mu, t)$ of basic context-sensitive usable rules of a term $t$ is defined as $R \setminus \{ g \mid f \rightarrow_{\mu} g \text{ for some } f \in DF\mu(t) \}$, where $\rightarrow_{\mu}$ is the transitive and reflexive closure of $\rightarrow$. If $P \subseteq P^1(F, X)$, then $U_B(R, \mu^2, P) = \bigcup_{t \rightarrow_r \in P} U_B(R, \mu^2, r)$.

Example 2. (Continuing Example 1) The cycles in the CS-DG are:

\[
\begin{align*}
\{ \text{SEL}(s(n), \text{cons}(x, x)) \} & \quad \text{(C1)} \\
\{ \text{MINUS}(s(x), s(y)) \rightarrow \text{MINUS}(x, y) \} & \quad \text{(C2)} \\
\{ \text{QUOTE}(s(x), s(y)) \rightarrow \text{QUOTE}(\text{MINUS}(x, y), s(y)) \} & \quad \text{(C3)}
\end{align*}
\]

Consider the cycle $C_3$; then, $U_B(R, \mu^2, C_3)$ contains the following rules:

\[
\begin{align*}
\text{minus}(x, 0) & \rightarrow x \\
\text{minus}(s(x), s(y)) & \rightarrow \text{minus}(x, y)
\end{align*}
\]

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 $R = \{ f(c(x), x) \rightarrow f(x, x), b \rightarrow c(b) \}$ [4] together with $\mu(f) = \{ 1, 2 \}$ and $\mu(c) = \emptyset$. Note that $(R, \mu)$ is conservative (and innermost $\mu$-terminating, see [4]).

We have a single cycle $C = \{ F(c(x), x) \rightarrow F(x, x) \}$. According to Definition 4, we have no usable rules because $F(x, x)$ contains no symbol in $F$. We could wrongly conclude $\mu$-termination of $(R, \mu)$, but we have the infinite minimal $(R, C, \mu^2)$-chain $F(c(b), b) \rightarrow F(b, b) \leftarrow F(c(b), b) \rightarrow \cdots$.

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 $(R, P)$-chains (or $P$-minimal sequences in [17]). In [14,17], an interpretation of terms as sequences of their possible reducts is used\(^2\). The definition of the transformation requires adding new fresh (list constructor) symbols $\perp, g \notin F$ and the (projection) rules $g(x, y) \rightarrow x$, $g(x, y) \rightarrow y$ (the $\pi$-rules). In this way, infinite minimal $(R, P)$-chains can be represented as infinite $(U(B(R, P)) \cup \pi, P)$-chains. We recall here the interpretation definition.

Definition 5 (Interpretation [14,17]). Let $R = (F, R)$ be a TRS and $G \subseteq F$. Let $> be an arbitrary total ordering over $T(F^* \cup \{ \perp, g \}, X)$ where $\perp$ is a new constant symbol and $g$ is a new binary symbol. The interpretation $I_G$ is a mapping

\(^2\) This method goes back to [15].
Usable Rules for Context-Sensitive Rewrite Systems

179

from terminating terms in \( T(\mathcal{F}^1, \mathcal{X}) \) to terms in \( T(\mathcal{F}^1 \cup \{ \bot, g \}, \mathcal{X}) \) defined as follows:

\[
I_\mu(t) = \begin{cases} 
  t & \text{if } t \in \mathcal{X} \\
  f(I_\mu(t_1), \ldots, I_\mu(t_n)) & \text{if } t = f(t_1 \ldots t_n) \text{ and } f \not\in \mathcal{G} \\
  g(f(I_\mu(t_1), \ldots, I_\mu(t_n)), t') & \text{if } t = f(t_1 \ldots t_n) \text{ and } f \in \mathcal{G}
\end{cases}
\]

where \( t' = \text{order}\left(\{I_\mu(u) \mid t \rightarrow_R u\}\right) \)

\[
\text{order}(T) = \begin{cases} 
  \bot, & \text{if } T = \varnothing \\
  g(t, \text{order}(T - \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } >
\end{cases}
\]

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 \( T(\mathcal{F}^1 \cup \{ \bot, g \}, \mathcal{X}) \).

Similarly, we aim at defining a \( \mu \)-interpretation \( I_{\mu, \mu} \) that allows us to associate an infinite \((\mu(\mathcal{R}, \mu^1, \mathcal{P}) \cup \pi, \mathcal{P}, \mu^1)\)-chain to each infinite minimal \((\mathcal{R}, \mathcal{P}, \mu^1)\)-chain. Actually, the main problem is that \((\mathcal{R}, \mathcal{P}, \mu^1)\)-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 \( I_{\mu, \mu} \) 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 \((\mathcal{R}, \mathcal{P}, \mu^1)\)-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, \( I_\mu \) (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 \((\mathcal{R}, \mathcal{P}, \mu^1)\)-chain at a \( \mu \)-replacing position, then \( t \) \( \geq \mu \) \( s \) and \( s^1 \) 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 \((\mu \text{-Interpretation})\). Let \( \mathcal{R} = (\mathcal{F}, \mu, \mathcal{R}) \) be a CS-TRS, \( \mathcal{G} \subseteq \mathcal{F} \) be such that \( \mathcal{G} \cap \mathcal{H} = \varnothing \). Let \( \pi \) be an arbitrary total ordering over \( T(\mathcal{F}^1 \cup \{ \bot, g \}, \mathcal{X}) \) where \( \bot \) is a new constant symbol and \( g \) is a new binary symbol (with \( \mu(g) = \{1, 2\} \)). The \( \mu \)-interpretation \( I_{\mu, \mu} \) is a mapping from arbitrary terms in \( T(\mathcal{F}^1, \mathcal{X}) \)
Usable Rules for Context-Sensitive Rewrite Systems 133
to terms in $T(F^\sharp \cup \{\bot, g\}, \mathcal{X})$ defined as follows:

$$I_{G,\mu}(t) = \begin{cases} 
  t & \text{if } t \in \mathcal{X} \\
  f(I_{G,\mu}(t_1), \ldots, I_{G,\mu}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin G \\
  g(f(I_{G,\mu}(t_1), \ldots, I_{G,\mu}(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin G \\
  \text{or } t \text{ is non-µ-terminating} \\
  \text{and } t \text{ is µ-terminating} 
\end{cases}$$

where
$$t' = \text{order}\left(\left\{I_{G,\mu}(u) \mid t \sim_{(R, \mu)} u\right\}\right)$$
order($T$) = \begin{cases} 
  \bot & \text{if } T = \emptyset \\
  g(t, \text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } >
\end{cases}

The set $G \subseteq F$ in Definition 6 corresponds to the set of non-usable symbols as discussed below. Now, we prove that $I_{G,\mu}$ 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 $I_{G,\mu}$ is well-defined both for $\mu$-terminating or non-$\mu$-terminating terms.

**Lemma 1.** Let $R = (F, R)$ be a TRS, $\mu \in M_F$ and let $G \subseteq F - H$. Then, $I_{G,\mu}$ 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 $R = \{a(x, y) \rightarrow b(x, x), d(x, e) \rightarrow a(x, x), b(x, c) \rightarrow d(x, x), c \rightarrow e\}$ with $\mu(a) = \mu(d) = \{1, 2\}, \mu(b) = \{1\}$ and $\mu(c) = \mu(e) = \emptyset$. The only cycle consists of the dependency pairs $C = \{a(x, y) \rightarrow b(x, x), d(x, e) \rightarrow a(x, x), B(x, c) \rightarrow d(x, x)\}$. 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 $c \rightarrow e$ as usable, we would assume $G = \{a, b, c, d, e\}$. Then, we cannot simulate the infinite minimal $(R, P, \mu^2)$-chain $A(c, c) \leftarrow B(c, c) \leftarrow D(c, c) \leftarrow A(c, c) \leftarrow \ldots$ because we have:

$$s = I_{G,\mu}(A(c, c)) = A(g(c, g(e, \bot)), g(c, g(e, \bot))) \rightarrow B(g(c, g(e, \bot)), g(c, g(e, \bot))) \rightarrow t$$

The interpreted term $g(c, g(e, \bot))$ 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 $B$ to obtain the term $B(g(c, g(e, \bot)), c)$ required for applying the next CS-DP $(B(x, c) \rightarrow d(x, x))$ which continues the previous $(R, P, \mu)$-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 side of the rules.

**Definition 7 (µ-Dependency).** Given a CS-TRS $R = (F, \mu, R)$, we say that $f \in F$ directly $\mu$-depends on $g \in F$, written $f \triangleright_{\mu} g$, 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.
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 $\text{NDFun}^\mu(t) = \{ f \mid \exists p \in \text{Pos}(t) \text{ and } p \not\in \text{Pos}(t), f = \text{root}(t[p]) \}$. 

**Definition 8 (Context-Sensitive Usable Rules).** Let $\mathcal{R} = (\mathcal{F}, \mathcal{R})$ be a TRS, $\mu \in M_\mathcal{R}$, and $\mathcal{P} \subseteq \mathcal{P}^d(\mathcal{F}, \mathcal{X})$. The set $\mathcal{U}(\mathcal{R}, \mu^d, \mathcal{P})$ of context-sensitive usable rules for $\mathcal{P}$ is given by $\mathcal{U}(\mathcal{R}, \mu^d, \mathcal{P}) = \bigcup_{i \in \mathcal{P}} \mathcal{U}_i(\mathcal{R}, \mu, l \rightarrow r)$, where $\mathcal{U}_i(\mathcal{R}, \mu, l \rightarrow r) = \mathcal{R} \setminus \{ g \mid f \circ_{\mu}^* g \text{ for some } f \in \text{DFun}^\mu(r) \cup \text{NDFun}^\mu(l) \}$.

Note that $\mathcal{U}_\mathcal{E}$ 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 $\mathcal{U}_\mathcal{E}(\mathcal{R}, \mu)$ the set of extended usable rules. On the other hand, $\mathcal{U}_\mathcal{F}$ 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 $\mathcal{R} = (\mathcal{F}, \mathcal{R})$ be a TRS, $\mathcal{P} \subseteq \mathcal{P}^d(\mathcal{F}, \mathcal{X})$, and $\mu \in M_\mathcal{R}$. If there exists a $\mu$-reduction pair $(\geq, \supseteq)$ such that $\mathcal{U}(\mathcal{R}, \mu^d, \mathcal{P}) \cup \pi \subseteq \geq, \mathcal{P} \subseteq \geq \cup \supseteq$, and

1. If $\mathcal{P}_X = \emptyset$, then $\mathcal{P} \cap \supseteq \neq \emptyset$,
2. If $\mathcal{P}_X \neq \emptyset$, then $\exists_\mu \subseteq \geq$, and
   - (a) $\mathcal{P} \cap \supseteq \neq \emptyset$ and $f(x_1, \ldots, x_k) \geq f^i(x_1, \ldots, x_k)$ for all $f^i \in \mathcal{P}$, or
   - (b) $f(x_1, \ldots, x_k) \supseteq f^i(x_1, \ldots, x_k)$ for all $f^i \in \mathcal{P}$.

Let $\mathcal{P}_\geq = \{ u \rightarrow v \in \mathcal{P} \mid u \supseteq v \}$. Then there are no infinite minimal $(\mathcal{R}, \mathcal{P}, \mu^d)$-chains whenever:

1. there are no infinite minimal $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_\supseteq, \mu^d)$-chains in case (1) and in case (2a),
2. there are no infinite minimal $(\mathcal{R}, (\mathcal{P} \setminus \mathcal{P}_X) \setminus \mathcal{P}_\supseteq, \mu^d)$-chains in case (2b).

**Proof (Sketch).** By contradiction. Assume that there exists an infinite minimal $(\mathcal{R}, \mathcal{P}, \mu^d)$-chain $\mathcal{A}$ but there is no infinite minimal $(\mathcal{R}, \mathcal{P} \setminus \mathcal{P}_\supseteq, \mu^d)$-chains in case (1) and (2a), or there is no infinite minimal $(\mathcal{R}, (\mathcal{P} \setminus \mathcal{P}_X) \setminus \mathcal{P}_\supseteq, \mu^d)$-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:

\[ t_1 \leftarrow_{\mathcal{R}, \mu} u_1 \rightarrow_\mathcal{P} \circ \triangleright_{\mu}^* t_2 \leftarrow_{\mathcal{R}, \mu} u_2 \rightarrow_\mathcal{P} \circ \triangleright_{\mu}^* \cdots \]

where $s \triangleright_{\mu}^* t$ for $s \in T(\mathcal{F}, \mathcal{X})$ and $t \in T^d(\mathcal{F}, \mathcal{X})$ means that $s \triangleright_{\mu}^* t^\sigma$.

Let $\sigma$ be a substitution, we denote by $\sigma_{\mathcal{R}, \mu}$ 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 $\mathcal{R} \setminus \mathcal{U}(\mathcal{R}, \mu^d, \mathcal{P})$. We show that after applying $I_{\mathcal{G}, \mu}$ we get an infinite $(\mathcal{U}(\mathcal{R}, \mu^d, \mathcal{P}) \cup \pi, \mathcal{P'}, \mu^d)$-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 $u_i \rightarrow P' \circ \sharp$ on $t$, we have two possibilities:
  1. There is $l \rightarrow r \in P'_n$, then we get:
     $$I_G \mu(u_i) \rightsquigarrow I_G \mu(l) \rightarrow P'_n \rightarrow I_G \mu(r) = I_G \mu(t_{i+1})$$
  2. There is an $l \rightarrow x \in P'_n$, then we get:
     $$I_G \mu(u_i) \rightsquigarrow I_G \mu(l) \rightarrow P'_n \rightarrow I_G \mu(x) = I_G \mu(x)$$

Therefore we get the infinite $(U(\mathcal{R}, \mu^2, \mathcal{P}), \mathcal{P}'_n)$-chain:

$$I_G \mu(t_1) \rightarrow I_G \mu(u_1) \rightarrow I_G \mu(u_1) \rightarrow \cdots$$

Using the premises of the theorem, by monotonicity and stability of $\geq$, we would have that $I_G \mu(t_i) \geq I_G \mu(u_i)$ for all $i \geq 1$. By stability of $\geq$ (and of $\leq$), we have that $I_G \mu(u_i) \geq I_G \mu(t_{i+1})$ for all $i \geq 1$ and $I_G \mu(u_i) \geq I_G \mu(t_{i+1})$ for all $j \in J$ for an infinite set $J = \{j_1, j_2, \ldots\}$ of natural numbers $j_1 < j_2 < \ldots < j_n < \ldots$. Now, since $\geq \circ \subseteq \lor \circ \geq \circ \subseteq \lor$, we would obtain an infinite sequence consisting of infinitely many $\lor$-steps. We obtain a contradiction to the well-foundedness of $\geq$. \qed

Remark 1. Notice that (as expected) $U(\mathcal{R}, \mathcal{P}, \mu^\tau) = U(\mathcal{R}, \mathcal{P})$, i.e., our usable rules for CS-TRSs $(\mathcal{R}, \mu)$ coincide with the standard definition (see Definition 2) when $\mu = \mu^\tau$ is considered (here, $\mu^\tau(f) = \{1, \ldots, ar(f)\}$ 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 $\geq_P$ of a reduction pair $(\geq_P, \lor_P)$ associated to a given set of pairs $\mathcal{P}$. We just need to consider $U(\mathcal{R}, \mu^2, \mathcal{P})$ (together with the $\pi$-rules).

Example 5. (Continuing Examples 1 and 2) Since $H \cap D = \{\text{from}, \text{zWquot}\}$, we have that $U(\mathcal{R}, \mu^2, C_1)$ is:

\[
\begin{align*}
\text{minus}(x, 0) & \rightarrow x & \text{minus}(s(x), s(y)) & \rightarrow \text{minus}(x, y) \\
\text{quote}(0, s(y)) & \rightarrow 0 & \text{quote}(s(x), s(y)) & \rightarrow s(\text{quote}(\text{minus}(x, y), s(y))) \\
\text{zWquot}(\text{nil}, x) & \rightarrow \text{nil} & \text{from}(x) & \rightarrow \text{cons}(x, \text{from}(s(x))) \\
\text{zWquot}(x, \text{nil}) & \rightarrow \text{nil} & \text{zWquot}(\text{cons}(x, x), s(y)) & \rightarrow \text{cons}(\text{quote}(x, y), \text{zWquot}(x, y))
\end{align*}
\]

According to Theorem 1, the following polynomial interpretation (computed by $\text{MU-term}[1, 21]$) shows the absence of infinite $(\mathcal{R}, C_1, \mu^2)$-chains.

\[
\begin{align*}
[s](x) & = x + 1 & \text{quote}(x, y) & = x + y & \text{minus}(x, y) & = 0 \\
[\text{from}(x)] & = 0 & [\text{sel}(x, y)] & = 0 & \text{zWquot}(x, y) & = x + y \\
[\text{cons}(x, y)] & = 0 & [0](x, y) & = 0 & \text{nil}(x, y) & = 1 \\
[\text{SEL}(x, y)] & = x
\end{align*}
\]
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 \( R \), it could be sometimes easier to find a proof of \( \mu \)-termination of the CS-TRS \( (R, \mu) \) 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 \( U_B(R, \mu, P) \) is contained in (and is usually smaller than) \( U(R, \mu, P) \), 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 \( F \) be a signature, \( \mu \in M_F \) and \( t \in T(F, X) \). We denote \( \text{Var}_\mu(t) \) the set of variables in \( t \) occurring at non-\( \mu \)-replacing positions, i.e., \( \text{Var}_\mu(t) = \{ x \in \text{Var}(t) \mid t \not\triangleright_\mu x \} \).

**Definition 10 (Strongly Conservative).** Let \( R \) be a TRS and \( \mu \in M_R \). A rule \( l \rightarrow r \) is strongly conservative if it is conservative and \( \text{Var}_\mu(l) \cap \text{Var}_\mu(l) = \text{Var}_\mu(r) \cap \text{Var}_\mu(r) = \emptyset \); and \( R \) is strongly conservative if all rules in \( R \) are strongly conservative.

Linear CS-TRSs trivially satisfy \( \text{Var}_\mu(l) \cap \text{Var}_\mu(l) = \text{Var}_\mu(r) \cap \text{Var}_\mu(r) = \emptyset \). 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 \((R, \mu, \mu^\mu)\)-chains, then we deal with \( \mu \)-terminating terms w.r.t. \((R, \mu)\). 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 \( I^\mu_{G, \mu} \) where non-\( \mu \)-replacing positions are not interpreted. In contrast to \( I^\mu_{G, \mu} \) (but closer to \( I^\mu_G \)) our new basic \( \mu \)-interpretation is defined now for \( \mu \)-terminating terms only.

**Definition 11 (Basic \( \mu \)-Interpretation).** Let \((F, \mu, R)\) be a CS-TRS and \( G \subseteq F \). Let \( \triangleright \) be an arbitrary total ordering over \( T(F^\dagger \cup \{ \bot, g \}, X) \) where \( \bot \) is a new constant symbol and \( g \) is a new binary symbol. The basic \( \mu \)-interpretation \( I^\mu_{G, \mu} \) is...
a mapping from \( \mu \)-terminating terms in \( T(F^\sharp, \mathcal{X}) \) to terms in \( T(F^\sharp \cup \{ \bot, g \}, \mathcal{X}) \) defined as follows:

\[
I^r_{\sigma, \mu}(t) = \begin{cases} 
  t & \text{if } t \in \mathcal{X} \\
  f(I^r_{\sigma, f, t_1}(t_1), \ldots, I^r_{\sigma, f, t_n}(t_n)) & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \notin G \\
  g(f(I^r_{\sigma, f, t_1}(t_1), \ldots, I^r_{\sigma, f, t_n}(t_n)), t') & \text{if } t = f(t_1, \ldots, t_n) \text{ and } f \in G
\end{cases}
\]

where \( I^r_{\sigma, \mu}(t) = \begin{cases} 
  I^\mu_{\mu}(t) & \text{if } i \in \mu(f) \\
  t & \text{if } i \notin \mu(f)
\end{cases} \)

\[t' = \text{order}(\{I^r_{\sigma, \mu}(u) \mid t \xrightarrow{\Rightarrow_{R, \mu}} u\})\]

\[\text{order}(T) = \begin{cases} 
  \bot, & \text{if } T = \emptyset \\
  \mu(f(t, \text{order}(T - \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } >
\end{cases}\]

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 \( I^r_{\sigma, \mu}(t) \) is finite.

For the proof of our next theorem, we need some auxiliary definitions and results.

**Definition 12.** Let \((R, \mu)\) be a CS-TRS and \( \sigma \) be a substitution and let \( G \subseteq F \). We denote by \( \sigma I^r_{\mu}: T(F, \mathcal{X}) \to T(F, \mathcal{X}) \) a function that, given a term \( t \) replaces occurrences of \( x \in \text{Var}(t) \) at position \( p \) in \( t \) by either \( I^r_{\sigma, \mu}(\sigma(x)) \) if \( p \in \text{Pos}^\mu(t) \), or \( \sigma(x) \) if \( p \notin \text{Pos}^\mu(t) \).

**Proposition 1.** Let \((R, \mu)\) be a CS-TRS and \( \sigma \) be a substitution and let \( G \subseteq F \). Let \( t \) be a term such that \( \text{Var}^\mu(t) \cap \text{Var}^\mu(t) = \emptyset \). Let \( \sigma I^r_{\mu, t} \) be a substitution given by

\[
\sigma I^r_{\mu, t}(x) = \begin{cases} 
  I^r_{\sigma, \mu}(\sigma(x)) & \text{if } x \in \text{Var}^\mu(t) \\
  \sigma(x) & \text{otherwise}
\end{cases}
\]

Then, \( \sigma I^r_{\mu, t}(t) = I^r_{\sigma, \mu}(t) \).

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 \( R = (F, R) \) be a TRS, \( P \subseteq \mathcal{P}^\sharp(F, \mathcal{X}) \), and \( \mu \in M_F \). If \( P \cup U\mathcal{D}(R, \mu^\sharp, P) \) is strongly conservative and there exists a \( \mu \)-reduction pair \((\geq, \sqsupseteq)\) such that \( U\mathcal{D}(R, \mu^\sharp, P) \cup P \subseteq \geq, P \subseteq \sqsupseteq, P \cap \sqsupseteq \neq \emptyset \). Let \( \mathcal{P} \supseteq \{ u \to v \in P \mid u \sqsupseteq v \} \). Then there are no infinite minimal \((R, P, \mu^\sharp)\)-chains whenever there are no infinite minimal \((R, P \setminus P_{\sqsupseteq}, \mu^\sharp)\)-chains.

**Proof (Sketch).** By contradiction. Assume that there exists an infinite minimal \((R, P, \mu^\sharp)\)-chain \( \mathcal{A} \) but there is no infinite minimal \((R, P \setminus P_{\sqsupseteq}, \mu^\sharp)\)-chains. We can assume that there is a \( P' \subseteq P \) such that \( \mathcal{A} \) has a tail \( B \) where all pairs are used infinitely often:

\[
i_1 \xrightarrow{R, \mu} u_1 \to P, i_2 \xrightarrow{R, \mu} u_2 \to P, \ldots
\]
After applying the basic \( \mu \)-interpretation \( I'_G,\mu \) we obtain an infinite \((U_B(R,\mu^\#),(P')\)-chain. Since all terms in the infinite \((R,\mu^\#,(P')\)-chain are \( \mu \)-terminating w.r.t. \((R,\mu)\), we can indeed apply the basic \( \mu \)-interpretation \( I'_G,\mu \).

Let \( i \geq 1 \).

- If we consider the pair step \( u_i \rightarrow_P t_{i+1} \) we can obtain the following sequence:

  \[
  I'_G,\mu(u_i) \xrightarrow[^{\sigma}]{\pi} I'_G,\mu(r) = I'_G,\mu(t_{i+1})
  \]

- If we consider the rewrite sequence \( t_i \rightarrow_P u_i \). All terms in it are \( \mu \)-terminating, then we get

  \[
  I'_G,\mu(t_i) \xrightarrow[^{\sigma}]{\pi} I'_G,\mu(u_i).
  \]

So we obtain the infinite \( \mu \)-rewrite sequence:

\[
I'_G,\mu(t_1) \xrightarrow[^{\sigma}]{\pi} I'_G,\mu(u_1) \xrightarrow[^{\sigma}]{\pi} \cdots
\]

Using the premise of the theorem, it is transformed into an infinite sequence consisting of \( \sqcup \) and infinitely many \( \sqcap \) steps. Using the stability condition, this contradicts the well-foundedness of \( \sqcap \). \( \square \)

**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 \((R,C_2,\mu^\#)\)-chains:

\[
[s](x) = x + 1 \quad \text{MINUS}(x,y) = y
\]

The basic usable rules \( U_B(R,\mu^\#,C_3) \) for \( C_3 \) are strongly conservative (see Example 2). The following polynomial interpretation proves the absence of infinite \((R,C_3,\mu^\#)\)-chains:

\[
[0] = 0 \quad [a](x) = x + 1 \quad \text{MINUS}(x,y) = x \quad \text{QUOT}(x,y) = x
\]

Since we dealt with cycle \( C_1 \) in Example 5, \( \mu \)-termination of \( R \) is proved. Until now, no tool for proving termination of CSR could find a proof for this \( R \) in Example 1. Thanks to the results in this paper, which have been implemented in \textsc{mu-term}, we can easily prove \( \mu \)-termination of \( R \) now.

**6 Experiments**

The techniques described in the previous sections have been implemented as part of the tool \textsc{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 \textsc{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 \textsc{mu-term} proves termination of CSR as termination
Table 1. Comparative among the three \textsc{mu-term} versions

<table>
<thead>
<tr>
<th>Tool Version</th>
<th>Proved</th>
<th>Total Time</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Usable Rules</td>
<td>44/90</td>
<td>6.11s</td>
<td>0.14s</td>
</tr>
<tr>
<td>Innermost Usable Rules</td>
<td>52/90</td>
<td>11.75s</td>
<td>0.23s</td>
</tr>
<tr>
<td>Usable Rules</td>
<td>64/90</td>
<td>8.91s</td>
<td>0.14s</td>
</tr>
</tbody>
</table>

Table 2. Comparative over the 44 examples

<table>
<thead>
<tr>
<th>Tool Version</th>
<th>Proved</th>
<th>Total Time</th>
<th>Average Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Usable Rules</td>
<td>44/90</td>
<td>6.11s</td>
<td>0.14s</td>
</tr>
<tr>
<td>Innermost Usable Rules</td>
<td>44/90</td>
<td>5.03s</td>
<td>0.11s</td>
</tr>
<tr>
<td>Usable Rules</td>
<td>44/90</td>
<td>3.57s</td>
<td>0.08s</td>
</tr>
</tbody>
</table>

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\(^3\). 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 \textsc{mu-term}, we see that version (3) of \textsc{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

\(^3\) See http://www.lri.fr/~marche/termination-competition/2007
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 M U - T E R M [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 M U - T E R M in winning the context-sensitive subcategory of the 2007 competition of termination tools.

References

8.10 Improving Context-Sensitive Dependency Pairs

Improving Context-Sensitive Dependency Pairs∗

Beatriz Alarcón1, Fabian Emmes2, Carsten Fuhs2, Jürgen Giesl2, Raúl Gutiérrez3, Salvador Lucas1, Peter Schneider-Kamp2, and René Thieemann3

1 DSIC, Universidad Politécnica de Valencia, Spain
2 LuFG Informatik 2, RWTH Aachen University, Germany
3 Institute of Computer Science, University of Innsbruck, Austria

Abstract. Context-sensitive dependency pairs (CS-DPs) are currently the most powerful method for automated termination analysis of context-sensitive rewriting. However, compared to DPs for ordinary rewriting, CS-DPs suffer from two main drawbacks: (a) CS-DPs can be collapsing. This complicates the handling of CS-DPs and makes them less powerful in practice. (b) There does not exist a “DP framework” for CS-DPs which would allow one to apply them in a flexible and modular way. This paper solves drawback (a) by introducing a new definition of CS-DPs. With our definition, CS-DPs are always non-collapsing and thus, they can be handled like ordinary DPs. This allows us to solve drawback (b) as well, i.e., we extend the existing DP framework for ordinary DPs to context-sensitive rewriting. We implemented our results in the tool AProVE and successfully evaluated them on a large collection of examples.

1 Introduction

Context-sensitive rewriting [23,24] models evaluations in programming languages. It uses a replacement map μ with μ(f) ⊆ {1,...,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)

\[
\begin{align*}
gt(0, y) &\rightarrow \text{false} & p(0) &\rightarrow 0 \\
gt(s(x), 0) &\rightarrow \text{true} & p(s(x)) &\rightarrow x \\
gt(s(x), s(y)) &\rightarrow gt(s(x), y) & \text{minus}(x, y) &\rightarrow \text{if}(gt(y, 0), \text{minus}(p(x), p(y)), x) \\
\text{if}(true, x, y) &\rightarrow x & \text{div}(0, s(y)) &\rightarrow 0 \\
\text{if}(false, x, y) &\rightarrow y & \text{div}(s(x), s(y)) &\rightarrow s(\text{div}(\text{minus}(x, y), s(y)))
\end{align*}
\]

with μ(if) = {1} and μ(f) = {1,...,arity(f)} for all other symbols f to model the usual behavior of if: in if(t1, t2, t3), one may evaluate t1, but not t2 or t3. It will turn out that due to μ, this CS-TRS is indeed terminating. In contrast, if one allows arbitrary reductions, then the TRS would be non-terminating.

∗ Authors from Valencia were partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grants TIN 2007-68093-C02-02 and HA 2006-0007. B. Alarcón was partially supported by the Spanish MEC/MICINN under FPU grant AP2005-3399. R. Gutiérrez was partially supported by the Spanish MEC/MICINN, under grant TIN 2004-7943-C04-02. Authors from Aachen were supported by the DAAD under grant D/06/12785 and by the DFG under grant GI 274/5-2.
There are two approaches to prove termination of context-sensitive rewriting. The first approach transforms CS-TRSs to ordinary TRSs, cf. [13,26]. But transformations often generate complicated TRSs where all termination tools fail.

Therefore, it is more promising to adapt existing termination techniques from ordinary term rewriting to the context-sensitive setting. Such adaptions were done for classical methods like RPO or polynomial orders [8,19,25]. However, much more powerful techniques like the \textit{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 \textit{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 \textit{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 \textit{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 \textit{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 \textit{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 \textsc{APoVE} [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 $\text{Pos}(s)$ be the set of \textit{positions} of a term $s$. For a replacement map $\mu$, we define the \textit{active positions} $\text{Pos}^\mu(s)$: For $x \in \mathcal{V}$ let $\text{Pos}^\mu(x) = \{\varepsilon\}$ where $\varepsilon$ is the root position. Moreover, $\text{Pos}^\mu(f(s_1, \ldots, s_n)) = \{\varepsilon\} \cup \{i.p \mid i \in \mu(f), p \in \text{Pos}^\mu(s_i)\}$. We say that $s \geq_t^\mu t$ holds if $t = s|_p$ for some $p \in \text{Pos}^\mu(s)$ and $s \geq_t^\mu t$ if $s \geq_t^\mu t$ and $s \neq t$. Moreover, $s \gg_t^\mu t$ if $t = s|_p$ for some $p \in \text{Pos}^\mu(s) \setminus \text{Pos}^\mu(s)$.

We denote the ordinary subterm relations by $\supset$ and $\triangleright$.

A CS-TRS $(\mathcal{R}, \mu)$ consists of a finite TRS $\mathcal{R}$ and a replacement map $\mu$. We have $s \leftarrow_{\mathcal{R}, \mu} t$ iff there are $\ell \rightarrow r \in \mathcal{R}$, $p \in \text{Pos}^\mu(s)$, and a substitution $\sigma$ with $s|_p = \sigma(\ell)$ and $t = s[\sigma(r)]|_{\mu}$. This reduction is an \textit{innermost} step (denoted $\leftarrow_{\mathcal{R}, \mu}$) if all $t$ with $s|_p \triangleright_{\mu} t$ are in normal form w.r.t. $(\mathcal{R}, \mu)$. A term $s$ is in \textit{normal form} w.r.t. $(\mathcal{R}, \mu)$ if there is no term $t$ with $s \leftarrow_{\mathcal{R}, \mu} t$. A CS-TRS $(\mathcal{R}, \mu)$ is \textit{terminating} if $\leftarrow_{\mathcal{R}, \mu}$ is well founded and \textit{innermost terminating} if $\leftarrow_{\mathcal{R}, \mu}$ is well founded.

minus(0, 0) →+ if (gt(0, 0), minus(0, 0), 0) →+ if (... if (gt(0, 0), minus(0, 0), 0), ...) →+ ...

Improving Context-Sensitive Dependency Pairs 191
Let $D = \{ \text{root}(\ell) \mid \ell \rightarrow r \in R \}$ be the set of defined symbols. For every $f \in D$, let $f^\uparrow$ be a fresh tuple symbol of same arity, where we often write "$f^\uparrow" instead of "\( f^\uparrow \)." For $t = f(t_1, \ldots, t_n) \in D$, let $t^\uparrow = f^\uparrow(t_1, \ldots, t_n)$.

**Definition 2 (CS-DPs [1])**. Let $(\mathcal{R}, \mu)$ be a CS-TRS. If $\ell \rightarrow r \in R$, and root($t$) $\in D$, then $t^\uparrow \rightarrow t^\uparrow$ is an ordinary dependency pair.\(^1\) If $t \rightarrow r \in R$, $\prod_{\mu x}$ for a variable $x$, and $t^\uparrow \rightarrow x$ is a collapsing DP. Let $D_P(\mathcal{R}, \mu)$ and $D_P(\mathcal{R}, \mu)$ be the sets of all ordinary resp. all collapsing DPs.

**Example 3.** For the TRS of Ex. 1, we obtain the following CS-DPs.

\[
\begin{align*}
\text{GT}(s(x), s(y)) &\rightarrow \text{GT}(x, y) \quad (2) \\
\text{IF}(\text{true}, x, y) &\rightarrow x \quad (3) \\
\text{IF}(\text{false}, x, y) &\rightarrow y \quad (4)
\end{align*}
\]

To prove termination, one has to show that there is no infinite chain of DPs. For ordinary rewriting, a sequence $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \ldots$, of DPs is a chain if there is a substitution $\sigma$ such that $t_i \sigma$ reduces to $s_{i+1} \sigma$.\(^2\) If all $t_i \sigma$ are terminating, then the chain is minimal [14,17,22]. But due to the collapsing DPs, the notion of "chains" has to be adapted when it is used with CS-DPs [1]. If $s_i \rightarrow t_i$ is a collapsing DP (i.e., if $t_i \in V$), then instead of $t_i \sigma \leftarrow_{\mathcal{R}, \mu} s_{i+1} \sigma$ (and termination of $t_i \sigma$ for minimality), one requires that there is a term $u_i$ with $t_i \sigma \mathrel{\prod_{\mu}} u_i$ and $w_i \rightarrow_{\mathcal{R}, \mu} s_{i+1} \sigma$. For minimal chains, $w_i$ must be terminating.

**Example 4.** Ex. 1 has the chain $(5), (3), (5)$ as $\text{IF}(\text{gt}(s(y), 0), \text{minus}(p(x), p(y)), x)$ $\leftarrow_{\mathcal{R}, \mu} \text{IF}(\text{true}, \text{minus}(p(x), p(s(y))), x)$ $\leftarrow_{(3), \mu} \text{minus}(p(x), p(s(y)))$ and $(\text{minus}(p(x), p(s(y))))^2 = M(p(x), p(s(y)))$ 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 $M(x, y) \rightarrow M(p(x), p(y))$. 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 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 $r \mathrel{\prod_{\mu}} x$, but $\ell \not\rightarrow_{\mu} x$. 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 from $\text{minus}(t_1, t_2)$ over if$(\ldots, \text{minus}(p(t_1), p(t_2)), \ldots)$

\(^1\) A refinement is to eliminate DPs where $\ell \not\rightarrow_{\mu} t$, cf. [1,9].

\(^2\) We always assume that different occurrences of DPs are variable-disjoint and consider substitutions whose domains may be infinite.
to 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.\footnote{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).}

**Definition 5 (Hidden Term).** Let (\(\mathcal{R}, \mu\)) be a CS-TRS. We say that \(t\) is a hidden term if root(\(t\)) \(\in \mathcal{D}\) and if there exists a rule \(r \rightarrow r' \in \mathcal{R}\) with \(r \triangleright \mu \, t\).

In Ex. 1, the only hidden term is 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

\[
\begin{align*}
  a &\rightarrow f(g(b)) & (9) \\
  f(x) &\rightarrow h(x) \\
  b &\rightarrow a
\end{align*}
\]

The CS-TRS has the following infinite rewrite sequence:

\[
\begin{align*}
  a &\rightarrow_{\mathcal{R}, \mu} f(g(b)) \rightarrow_{\mathcal{R}, \mu} h(g(b)) \rightarrow_{\mathcal{R}, \mu} g(b) \rightarrow_{\mathcal{R}, \mu} g(a) \rightarrow_{\mathcal{R}, \mu} \ldots
\end{align*}
\]

We obtain the following CS-DPs according to Def. 2:

\[
\begin{align*}
  A &\rightarrow F(g(b)) & H(x) &\rightarrow x & (10) \\
  F(x) &\rightarrow H(x) \\
  B &\rightarrow A
\end{align*}
\]

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 \(g(\Box)\) a hiding context, since it can “hide” a hidden term. Moreover, the composition of hiding contexts is again a hiding context.

**Definition 7 (Hiding Context).** Let (\(\mathcal{R}, \mu\)) be a CS-TRS. The function symbol \(f\) hides position \(i\) if there is a rule \(r \rightarrow r' \in \mathcal{R}\) with \(r \triangleright \mu \, f(r_1, \ldots, r_i, \ldots, r_n)\), \(i \in \mu(f)\), and \(r_i\) contains a defined symbol or a variable at an active position. A context \(C\) is hiding if \(C = \Box\) or \(C\) has the form \(f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_n)\) where \(f\) hides position \(i\) and \(C'\) is a hiding context.

**Example 8.** In Ex. 6, \(g\) hides position \(1\) due to Rule (9). So the hiding contexts are \(\Box, g(\Box), g(g(\Box)), \ldots\). In the TRS of Ex. 1, minus hides both positions 1 and 2 and \(p\) hides position 1 due to Rule (1). So the hiding contexts are \(\Box, p(\Box), \text{minus}(\Box, \Box), p(p(\Box)), \text{minus}(\Box, p(\Box)), \ldots\).
on et al. ∅
U P → i t chains in the following, we included the “minimality
Alternatively, one could also use different
B for every
P i s with n t s and the following
DP → → μ s for all (U U R
Since we only regard
DP and
iff DP (R
μ (U) = ∅, if DP (R
μ (U) = ∅). Then the set of improved CS-DPs is DP(R, μ) = DP(μ) ∪ DP(μ, R).
Example 10. In Ex. 6, instead of (10) we get the unhiding DPs
H(x) → U(x), U(g(x)) → U(x), U(b) → B.
Now there is indeed an infinite chain. In Ex. 1, instead of (3) and (4), we obtain:5
IF(true, x, y) → U(x) (11)
IF(false, x, y) → U(y) (12)
U(minus(p(x), p(y))) → M(p(x), p(y)) (13)
U(p(x)) → P(x) (14)
U(p(x)) → U(x) (15)
U(minus(x, y)) → U(x) (16)
U(minus(x, y)) → U(y) (17)
Clearly, the improved CS-DPs are never collapsing. Thus, now the definition of (minimal)6 chains is completely analogous to the one for ordinary rewriting.
Definition 11 (Chain). Let P and R be TRSs and let μ be a replacement map. We extend μ to tuple symbols by defining μ(f) = μ(f) for all f ∈ D and μ(U) = ∅.7 A sequence of pairs s1 → t1, s2 → t2, . . . from P is a (P, R, μ)-chain iff there is a substitution σ with tσ ←− R,μ sσ and tσ is terminating w.r.t. (R, μ) for all i. It is an innermost (P, R, μ)-chain iff tσ ←− R,μ σi σ and σ is in normal form, and tσ is innermost terminating w.r.t. (R, μ) 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 (R, μ) is terminating iff there is no infinite (DP(R, μ), R, μ)-chain and innermost terminating iff there is no infinite innermost (DP(R, μ), R, μ)-chain.
Proof. We only prove the theorem for “full” termination. The proof for innermost termination is very similar and can be found in [5].

" Alternatively, one could also use different U-symbols for different collapsing DPs.
5 We omitted the DP U(p(y)) → P(y) that is “identical” to (14).
6 Since we only regard minimal chains in the following, we included the “minimality requirement” in Def. 11, i.e., we require that all tσ are (innermost) terminating.
As in the DP framework for ordinary rewriting, this restriction to minimal chains is needed for several DP processors (e.g., for the reduction pair processor of Thm. 21).
7 We define μ(U) = ∅, 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

\( \mathcal{M}_{\omega, \mu} \) contains all minimal non-terminating terms: \( t \in \mathcal{M}_{\infty, \mu} \) iff \( t \) is non-terminating and every \( r \) with \( t \triangleright_{\mu} s \triangleright_{\mu} r \) terminates. A term \( u \) has the hiding property if

- \( u \in \mathcal{M}_{\omega, \mu} \) and
- whenever \( u \triangleright_{\mu} s \triangleright_{\mu} t \) for some terms \( s \) and \( t \) with \( t \in \mathcal{M}_{\omega, \mu} \), then \( t \) is an instance of a hidden term and \( s = C[t'] \) for some hiding context \( C \).

We first prove the following claim:

Let \( u \) be a term with the hiding property and let \( u \triangleleft_{\mathcal{R}, \mu} v \triangleright_{\mu} w \) with \( w \in \mathcal{M}_{\infty, \mu} \). Then \( w \) also has the hiding property.

Let \( w \triangleright_{\mu} s \triangleright_{\mu} t' \) for some terms \( s \) and \( t' \) with \( t' \in \mathcal{M}_{\omega, \mu} \). Clearly, this also implies \( v \triangleright_{\mu} s \). If already \( u \triangleright_{\mu} s \), 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, \( u \not\triangleright_{\mu} s \). There must be a rule \( \ell \triangleright_{\mu} r \in \mathcal{R} \), 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, \( u \not\triangleright_{\mu} s \) implies \( \delta(\ell) \not\triangleright_{\mu} D \). Hence, \( v \triangleright_{\mu} s \) means \( \delta(r) \triangleright_{\mu} s \). (The root of \( s \) cannot be above \( \Box \) in \( D \) since those positions would be active.) Note that \( s \) cannot be at or below a variable position of \( r \), because this would imply \( \delta(\ell) \triangleright_{\mu} s \). Thus, \( s \) is an instance of a non-variable subterm of \( r \) (i.e., an inactive position). So there is a \( r' \not\in \mathcal{V} \) with \( r \triangleright_{\mu} r' \) and \( s = \delta(\ell') \). Recall that \( s \triangleright_{\mu} t' \), i.e., there is a \( p \in \text{Pos}^R(s) \) 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 \( t' \in \mathcal{M}_{\omega, \mu} \) implies root(\( t' \) ) \( \in D \)). Hence, \( r' |_p \) is a hidden term and thus, \( t' \) is an instance of a hidden term. Moreover, any instance of the context \( C' = r'(\Box) \) is hiding. If we define \( C \) to be \( \delta(C') \), \( t' = \delta(\ell') = \delta(r') |_{p} = \delta(C') |_{p} = C' |_{p} \) 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, \( \delta(x) |_{p_1} = t' \)). Since \( x \) also occurs in \( \ell \), we have \( \delta(\ell) \triangleright_{\mu} \delta(x) \) and thus \( u \triangleright_{\mu} \delta(x) \). Due to the minimality of \( u \) this implies \( u \triangleright_{\mu} \delta(x) \). 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 \( \delta(x) = C[t'] \) for a hiding context \( C \). Note that since \( r' |_{p_1} \) is a variable, the context \( C' \) around this variable is also hiding (i.e., \( C' = r'(\Box) \)). Thus, the context \( C = \delta(C') |_{C} \) is hiding as well and \( s = \delta(r') = \delta(r') \delta(\ell(x)) |_{p_2} = \delta(C') |_{C} |_{p_2} = C' |_{p_2} \).

Proof of Thm. 12 using Claim (18)

If \( \mathcal{R} \) is not terminating, then there is a \( t \in \mathcal{M}_{\omega, \mu} \) that is minimal w.r.t. \( \triangleright_{\mu} \). So there are \( t, t_1, t_2, t_3, \ldots \) such that

\[
t \xrightarrow{\omega} \mathcal{R}, \mu t_1 \xrightarrow{\omega} \mathcal{R}, \mu t_2 \xrightarrow{\omega} \mathcal{R}, \mu t_3 \xrightarrow{\omega} \mathcal{R}, \mu t_4 \ldots
\]

where \( t, t_i \in \mathcal{M}_{\omega, \mu} \) and all proper subterms of \( t \) (also at inactive positions) terminate. Here, \( \omega \) (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 $t \xrightarrow{\sigma \rightarrow_w} t_1$ 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 $t_i^2 \xrightarrow{\sigma \rightarrow_w} t_i^2$ and that all terms in the reduction $t_i^2 \xrightarrow{\sigma \rightarrow_w} t'_{i+1}$ terminate w.r.t. $(\mathcal{R}, \mu)$. As $t_i^2 \xrightarrow{\sigma \rightarrow_w} t_i^2$, we get an infinite $(\mathcal{D}(\mathcal{R}, \mu), \mathcal{R}, \mu)$-chain.

From (19) we know that there are $\ell_i \rightarrow r_i \in \mathcal{R}$ and $p_i \in \mathcal{P}(\mathcal{R}(s_i))$ with $t_i = \ell_i \sigma, s_i = r_i \sigma$, and $s_i|_{p_i} = r_i|_{p_i} = t'_{i+1}$ for all $i$. First let $p_i \in \mathcal{P}(\mathcal{R}(s_i))$ with $r_i|_{p_i} \notin \mathcal{U}$. Then $t_i^2 \rightarrow (r_i|_{p_i})^2 \in \mathcal{D}(\mathcal{R}, \mu)$ and $t_i^2 = t_i^2 \sigma \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} (r_i|_{p_i})^2 \sigma = t'_{i+1}^2$. Moreover, as $t_i, t'_{i+1} \in \mathcal{M}_{\infty, \mu}$, the terms $t_i^2$ and $t'_{i+1}^2$ 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 $t_i$. Thus, $t_i^2 \rightarrow U(x_i) \in \mathcal{D}(\mathcal{R}, \mu)$ and $r_i = C_i[x]$ where $C_i$ is an active context. Recall that $t_i = t_i \sigma$ has the hiding property and that $t_i \sigma \rightarrow_w t'_i \sigma \rightarrow_w t'_{i+1}$. Thus, we have $\sigma(x_i) = C_i[t'_{i+1}]$ for a hiding context $C_i$ and moreover, $t'_i$ is an instance of a hidden term. Hence we obtain:

$$t_i^2 \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(\sigma(x_i)) = U(C_i[t'_{i+1}])$$

$$t_i^2 \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(\sigma(x_i)) = U(C_i[t'_{i+1}])$$

$$t_i^2 \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(\sigma(x_i)) = U(C_i[t'_{i+1}])$$

$$t_i^2 \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(\sigma(x_i)) = U(C_i[t'_{i+1}])$$

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 $t_i^2$ and $t'_{i+1}^2$ are terminating. Moreover, all terms $U(\ldots)$ are normal forms since $\mu(U) = \emptyset$ and since $U$ does not occur in $\mathcal{R}$.

Completeness

Let there be an infinite chain $v_1 \rightarrow v_2 \rightarrow v_3 \rightarrow \ldots$ of improved CS-DPs. First, let the chain have an infinite tail consisting only of DPs of the form $U(f(x_1, \ldots, x_n)) \rightarrow U(x_k)$. Since $\mu(U) = \emptyset$, there are terms $t_i$ with $U(t_i) \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(t_2) \rightarrow_{\mathcal{D}(\mathcal{R}, \mu)} U(t_3) \ldots$. Hence, $t_i \rightarrow U(t_2) \rightarrow U(t_3)$ which contradicts the well-foundedness of $\rightarrow_w$.

Now we regard the remaining case. Here the chain has infinitely many DPs $v \rightarrow w$ with $v = t_i^2$ 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 $v_i^2 \sigma \rightarrow_{\mathcal{R}, \mu} C[w_i^2 \sigma]$ for an active context $C$. Here, $(f^2(t_1, \ldots, t_n)^2) = f(t_1, \ldots, t_n)$ for all $f \in \mathcal{D}$. Doing this for all such DPs implies that there is an infinite reduction w.r.t. $(\mathcal{R}, \mu)$.

If $v_i \rightarrow w_i \in \mathcal{D}(\mathcal{R}, \mu)$ then the claim is trivial, because then $j = i + 1$ and $v_i \sigma \rightarrow_{\mathcal{R}, \mu} C[w_i^2 \sigma] \rightarrow_{\mathcal{R}, \mu} C[w_i^2 \sigma]$ for some active context $C$.

Otherwise, $v_i \rightarrow w_i$ has the form $v_i \rightarrow U(x)$. Then $v_i \sigma \rightarrow_{\mathcal{R}, \mu} C[\sigma(x)]$ for an active context $C$. Moreover, $U(\sigma(x))$ reduces to $U(\delta(t))$ for a hidden term $t$ and
Improving Context-Sensitive Dependency Pairs

By removing hiding contexts. Since hiding contexts are active, \( \sigma(x) = C_2[\delta(t)] \) for an active context \( C_2 \). Finally, \( t\delta \overset{\rightarrow}{\longrightarrow}_{R,\mu} v_j\sigma \) and thus, \( t\delta \overset{\rightarrow}{\longrightarrow}_{R,\mu} v_i^\sigma \). By defining \( C = C_1[C_2] \), we get \( v_i^\sigma \overset{\rightarrow}{\longrightarrow}_{R,\mu} C[v_i^\sigma] \).

4 CS Dependency Pair Framework

By Thm. 12, (innermost) termination of a CS-TRS is equivalent to absence of infinite (innermost) chains. For ordinary rewriting, the DP framework is the most recent and powerful collection of methods to prove absence of infinite chains automatically. Due to our new notion of (non-collapsing) CS-DPs, adapting the DP framework to the context-sensitive case now becomes much easier. For this reason, we omitted the proofs in this section and refer to [5] for all proofs.

Definition 13 (CS-DP Problem and Processor). A CS-DP problem is a tuple \((P, R, \mu, e)\), where \( P \) and \( R \) are TRSs, \( \mu \) is a replacement map, and \( e \in \{t, i\} \) is a flag that stands for termination or innermost termination. We also call \((P, R, \mu, t)\)-chains “\((P, R, \mu, t)\)-chains” and we call innermost \((P, R, \mu)\)-chains “\((P, R, \mu, i)\)-chains”. A CS-DP problem \((P, R, \mu, e)\) is finite if there is no infinite \((P, R, \mu, e)\)-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 \((R, \mu)\), the termination proof starts with the initial DP problem \((\text{DP}(R, \mu), R, \mu, e)\) 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 \( e = i \) whenever possible. As shown in [12], termination and innermost termination coincide for CS-TRSs \((R, \mu)\) where \( R \) is orthogonal (i.e., left-linear and without critical pairs). So \((\text{DP}(R, \mu), R, \mu, i)\) 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 \((P, R, \mu, e)\), the nodes of the \((P, R, \mu, e)\)-dependency graph are the pairs of \( P \), and there is an arc from \( v \rightarrow w \) to \( s \rightarrow t \) iff \( v \rightarrow w, s \rightarrow t \) is a \((P, R, \mu, e)\)-chain.

For this reason, we omitted the proofs in this section and refer to [5] for all proofs.
Example 15. Fig. 1 shows the dependency graph for Ex. 1, for both $e \in \{ t, i \}$.9

A set $P' \neq \emptyset$ of DPs is a cycle if for every $v \rightarrow w$, $s \rightarrow t \in P'$, there is a non-empty path from $v \rightarrow w$ to $s \rightarrow t$ traversing only pairs of $P'$. A cycle $P'$ is a strongly connected component ("SCC") if $P'$ is not a proper subset of another cycle.

One can prove termination separately for each SCC. Thus, the following processor (whose soundness is obvious and completely analogous to the non-context-sensitive case) modularizes termination proofs.

Theorem 16 (CS-Dependency Graph Processor). For $d = (P, R, \mu, e)$, let $Proc(d) = \{(P_1, R, \mu, e), \ldots, (P_n, R, \mu, e)\}$, where $P_1, \ldots, P_n$ are the SCCs of the $(P, R, \mu, e)$-dependency graph. Then $Proc$ is sound.

Example 17. The graph in Fig. 1 has the three SCCs $P_1 = \{(2)\}, P_2 = \{(7)\}, P_3 = \{(5), (11)-(13), (15)-(17)\}$. Thus, the initial DP problem $(DP(P, R, \mu), R, \mu, I)$ is transformed into the new problems $(P_1, R, \mu, I), (P_2, R, \mu, I), (P_3, R, \mu, I)$.

As in the non-context-sensitive setting, the CS-dependency graph is not computable and thus, one has to use estimations to over-approximate the graph. For example, [1,4] adapted the estimation of [6] that was originally developed for ordinary rewriting: $Cap_{P'}^\mu(t)$ replaces all active subterms of $t$ with defined root symbol by different fresh variables. Multiple occurrences of the same such subterm are also replaced by pairwise different variables. $Ren_{P'}^\mu(t)$ replaces all active occurrences of variables in $t$ by different fresh variables (i.e., no variable occurs at several active positions in $Ren_{P'}^\mu(t)$). So $Ren_{P'}^\mu(Cap_{P'}^\mu([IF(g(y, 0), \text{min}(p(x), p(y))), x])] = Ren_{P'}^\mu(IF(z', \text{min}(p(x), p(y))), x]) = IF(x', \text{min}(p(x), p(y)), x)$.

To estimate the CS-dependency graph in the case $e = t$, one draws an arc from $v \rightarrow w$ to $s \rightarrow t$ whenever $Ren_{P'}^\mu(Cap_{P'}^\mu(w))$ and $s$ unify.10 If $e = i$, then one can modify $Cap_{P'}^\mu$ and $Ren_{P'}^\mu$ by taking into account that instantiated subterms at active positions of the left-hand side must be in normal form, cf. [4]. $Cap_{P'}^\mu(w)$ is like $Cap_{P'}^\mu(w)$, but the replacement of subterms of $w$ by fresh variables is not done if the subterms also occur at active positions of $v$. Similarly, $Ren_{P'}^\mu(w)$ is like $Ren_{P'}^\mu(w)$, but the renaming of variables in $w$ is not done if the variables

9 To improve readability, we omitted nodes (6) and (14) from the graph. There are arcs from the nodes (8) and (13) to (6) and from all nodes (11), (15), (16), (17) to (14). But (6) and (14) have no outgoing arcs and thus, they are not on any cycle.

10 Here (and also later in the instantiation processor of Sect. 4.3), we always assume that $v \rightarrow w$ and $s \rightarrow t$ are renamed apart to be variable-disjoint.
also occur active in $v$. Now we draw an arc from $v \rightarrow w$ to $s \rightarrow t$ whenever $\text{Ren}^e_\mu(\text{Cap}^e_\mu(w))$ and $s$ unify by an mgu $\theta$ where $e \theta$ and $s \theta$ are in normal form.\footnote{These estimations can be improved further by adapting existing refinements to the context-sensitive case. However, different to the non-context-sensitive case, for $e = 1$ it is not sufficient to check only for unification of $\text{Cap}^e_\mu(w)$ and $s$ (i.e., renaming variables with $\text{Ren}^e_\mu$ is also needed). This can be seen from the non-innermost terminating CS-TRS $(R, \mu)$ from [4, Ex. 8] with $R = \{f(s(x), x) \rightarrow f(x, x), a \rightarrow s(a)\}$ and $\mu(t) = \{1\}, \mu(s) = \emptyset$. Clearly, $\text{Cap}^e_{f(s(x), x)}(F(x, x)) = F(x, x)$ does not unify with $F(s(y), y)$. In contrast, $\text{Ren}^e_{f(s(x), x)}(\text{Cap}^e_{f(s(x), x)}(F(x, x))) = F(x, x)$ unifies with $F(s(y), y)$. Thus, without using $\text{Ren}^e_{f(s(x), x)}$ one would conclude that the dependency graph has no cycle and wrongly prove (innermost) termination.}

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 $(\succ, \succ)$ [1]. Here, $\succeq$ is a stable $\mu$-monotonic quasi-order, $\succ$ is a stable well-founded order, and $\succeq$ and $\succ$ are compatible (i.e., $\succ \circ \succeq \succeq$ or $\succeq \circ \succ \succeq$). Here, $\mu$-monotonicity means that $s_i \succeq t_i$ implies $f(s_1, \ldots, s_i, \ldots, s_n) \succeq f(s_1, \ldots, t_i, \ldots, s_n)$ whenever $i \in \mu(f)$.

For a DP problem $(P, R, \mu, e)$, the generated constraints ensure that some rules in $P$ are strictly decreasing (w.r.t. $\succ$) and all remaining rules in $P$ and $R$ are weakly decreasing (w.r.t. $\succeq$). Requiring $\ell \succeq r$ for all $\ell \rightarrow r \in R$ ensures that in a chain $s_1 \rightarrow t_1, s_2 \rightarrow t_2, \ldots$ with $t_i \sigma \succ \mu_{\rightarrow} s_{i+1} \sigma$, we have $t_i \sigma \succeq s_{i+1} \sigma$ for all $i$. Hence, if a reduction pair satisfies the constraints, then one can delete the strictly decreasing pairs from $P$ 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 $\succeq$ is $\mu$-compatible, i.e., $c(x,y) \succeq x$ and $c(x,y) \succeq y$ 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 $f \triangleright_{R} h$) then the $h$-rules are usable as well. Here, $f \triangleright_{R} h$ if $f = h$ or if there is a symbol $g$ with $g \triangleright_{R} h$ 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.
As in [4,20], Def. 18 adapts\textsuperscript{12} the concept of usable rules to the CS setting, resulting in $\mathcal{U}^\bullet (P,R,\mu)$. 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 $\mathcal{F}^\bullet(t)$ (resp. $\mathcal{F}^f(t)$) contain all function symbols occurring at active (resp. inactive) positions of a term $t$.

**Definition 18 (CS-Usable Rules).** Let $Rls(f) = \{ t \rightarrow r \in R \mid \text{root}(t) = f \}$. For any symbols $f,h$ and CS-TRS $(\mathcal{R},\mu)$, 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 $g \in \mathcal{F}^\bullet(\ell)$. 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 $g \in \mathcal{F}^f(\ell) \setminus \mathcal{F}(r)$. We define two forms of usable rules:

$\mathcal{U}^ \bullet (P,R,\mu) = \bigcup_{s,t \in P, f \in \mathcal{F}^\bullet(\ell), f \triangleright_{\mathcal{R},\mu} s} \text{Rls}(s)$

$\mathcal{U}^f (P,R,\mu) = \bigcup_{s,t \in P, f \in \mathcal{F}^f(\ell), f \triangleright_{\mathcal{R},\mu} s} \text{Rls}(s) \cup \bigcup_{\ell, r \in R, f \in \mathcal{F}(r), f \triangleright_{\mathcal{R},\mu} s} \text{Rls}(s)$

**Example 19.** We continue Ex. 17. $\mathcal{U}^\bullet (P_2, R, \mu) = \emptyset$ for $P_2 = \{1\}$, since there is no defined symbol at an active position in the right-hand side $\text{GT}(x,y)$ of (2). For $P_3 = \{5\}$, $\mathcal{U}^\bullet (P_3, R, \mu)$ are the minus-, if-, and $\text{gt}$-rules, since minus occurs at an active position in $\text{D}(\text{min}(x,y),s(y))$ and minus depends on if and gt. For $P_3 = \{5\}$, $\mathcal{U}^f (P_3, R, \mu)$ are the $\text{gt}$- and $p$-rules, as $gt$ and $p$ are the only defined symbols at active positions of right-hand sides in $P_3$.

In contrast, all $\mathcal{U}^\bullet (P_3, R, \mu)$ 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., $\mathcal{U}^\bullet (P,R,\mu)$) can only be used for conservative CS-TRSs (if $e = 1$) resp. strongly conservative CS-TRSs (if $e = t$).\textsuperscript{13} Let $\mathcal{V}^\bullet(t)$ (resp. $\mathcal{V}^f(t)$) 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 $\mathcal{V}^\bullet(r) \subseteq \mathcal{V}^\bullet(t)$ for all rules $t \rightarrow r \in R$. It is strongly conservative iff it is conservative and moreover, $\mathcal{V}^\bullet(t) \cap \mathcal{V}^f(t) = \emptyset$ and $\mathcal{V}^f(t) \cap \mathcal{V}^f(r) = \emptyset$ for all rules $t \rightarrow r \in R$.

Now we can define the reduction pair processor.

**Theorem 21 (CS-Reduction Pair Processor).** Let $(\succeq,\succ)$ be a $\mu$-reduction pair. For a CS-DP Problem $d = (P, R, \mu, e)$, the result of $\text{Proc}(d)$ is

\begin{itemize}
  \item $\{(P \setminus \succ, \mathcal{R}, \mu, e)\}$, if $P \subseteq (\succ \cup \succeq)$ and at least one of the following holds:
\end{itemize}

\textsuperscript{12}The adaptations can also be extended to refined definitions of usable rules [15,17].

\textsuperscript{13}The corresponding counterexamples in [4,20] show that these restrictions are still necessary for our new notion of CS-DPs. In cases where one cannot use $\mathcal{U}^\bullet$, 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 $\mathcal{U}^\bullet$ 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.
where we use Thm. 21 (iii)

\[ \mu \subseteq n \langle P, R, \mu \rangle \]

is strongly conservative, \( \geq \) is \( \mathcal{L}_\mu \)-compatible

(iv) \( \mu \subseteq n \langle P, R, \mu \rangle \)

is \( \mathcal{L}_\mu \)-compatible, \( e = 1 \)

(v) \( \mu \subseteq n \langle P, R, \mu \rangle \)

is \( \mathcal{L}_\mu \)-compatible

(vi) \( \mu \subseteq n \langle P, R, \mu \rangle \)

is \( \mathcal{L}_\mu \)-compatible

Then Proc is sound.

Example 22. As \( \mu \subseteq n \langle P, \emptyset, \mu \rangle \) is even strongly conservative, by Thm. 21 (i) or (ii) we only have to orient (2), which already works with the embedding order. So \( (P_1, R, \mu, 1) \) is transformed to the empty set of DP problems.

For \( P_2 = \{ (7) \} \), \( \mu \subseteq n \langle P_2, R, \mu \rangle \) 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]:

\[ D(x, y) = \text{max}(x, y) = x + 1, \text{if}(x, y, z) = \text{max}(y, z), \text{gt}(x, y) = \text{true} = \text{false} = 0. \]

Then the DP (7) is strictly decreasing and all rules from \( \mathcal{U}^\mu(P_2, R, \mu) \) are weakly decreasing.

Thus, the processor also transforms \( (P_2, R, \mu, 1) \) to the empty set of DP problems.

Finally, we regard \( P_3 = \{ (5), (11)-(13), (15)-(17) \} \) where we use Thm. 21 (iii) with the interpretation \( M(x, y) = \text{min}(x, y) = x + 1, \text{if}(x, y, z) = \text{max}(y, z), \text{gt}(x, y) = \text{true} = \text{false} = 0. \)

Next we apply \( M(x, y) = \text{min}(x, y) = x + 1, \text{if}(x, y, z) = \text{max}(y, z), \text{gt}(x, y) = \text{true} = \text{false} = 0. \)

Now (12) is strictly decreasing and all other remaining DPs and usable rules are weakly decreasing. Removing (12) yields \( (\{ (5), (11), (13), (15) \}, R, \mu, 1) \).

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 \( \geq \) (i.e., \( \mu \subseteq \geq \)) 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). With our new non-collapsing CS-DPs, this requirement is no longer needed.

Example 23. If one requires \( \geq \mu \subseteq \geq \), then the reduction processor would fail for Ex. 1, since then one cannot make the DP (7) strictly decreasing. The reason is that due to \( 2 \in \mu(\text{minus}) \), \( \mu(\text{gt}) \subseteq \geq \) implies \( \text{min}(x, y) \geq y \). So one cannot “filter away” the second argument of minus. But then a strict decrease of DP (7) together with \( \mu \)-monotonicity of \( \geq \) implies \( \text{D}(s(x), s(s(x))) \supset \text{D}(\text{min}(x, s(x))), s(s(x))) \supset \text{D}(s(x), s(s(x))), \) in contradiction to the well-foundedness of \( \supset \).

Moreover, previous CS-reduction pair processors also require \( f_1(x_1, \ldots, x_n) \geq f_1(x_1, \ldots, x_n) \) for all \( f \in D \) or \( f(x_1, \ldots, x_n) \supset f_1(x_1, \ldots, x_n) \) for all \( f \in D \). This requirement also destroys an important feature of the DP method, i.e., that tuple symbols \( f \) 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 DP’s [17]). We now adapt the instantiation processor to the context-sensitive setting. Similar adaptions can also be done for the other processors.15

The idea of this processor is the following. For a DP \( s \rightarrow t \), 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 \( \text{Ren}^\nu(\text{Cap}^\nu(w)) \) and \( s \) if \( e = t \) and analogously for \( e = i \).16 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 \( \ldots , v \rightarrow w , s \rightarrow t , \ldots \) 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 \( \mathcal{P}' = \mathcal{P} \cup \{ s \rightarrow t \} \). 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

\[
\begin{align*}
\overline{\mathcal{P}} &= \{ s \rightarrow t \mid \exists \mu (\text{Ren}^\nu(\text{Cap}^\nu(w)), s), v \rightarrow w \in \mathcal{P}' \}, \text{ if } e = t \\
\overline{\mathcal{P}} &= \{ s \rightarrow t \mid \exists \mu (\text{Ren}^\nu(\text{Cap}^\nu(w)), s), v \rightarrow w \in \mathcal{P}', \mu, e \text{ normal}, \mu = i \}
\end{align*}
\]

Then \( \text{Proc} \) is sound.

Example 25. For the TRS of Ex. 1, we still had to solve the problem \((\{ 5 \}, \{ 11 \}, \{ 13 \}, \{ 15 \}), \mathcal{R}, \mu, i)\), 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 \( \text{Ren}^\nu(\text{Cap}^\nu(\text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x))) = \text{IF}(z', \text{minus}(p(x), p(y)), x) \), cf. Sect. 4.1. So the mgu is \( \theta = [ z'/\text{true}, x'/\text{minus}(p(x), p(y)) ] \). Hence, we can replace (11) by

\[
\text{IF}(\text{true}, \text{minus}(p(x), p(y))), x) \rightarrow \text{U}(\text{minus}(p(x), p(y)))
\]

where the CS variant of the instantiation processor is advantageous over the non-CS one which uses \( \text{Cap} \) instead of \( \text{Cap}^\nu \), where \( \text{Cap} \) 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 \( \mu(U) = \emptyset \). So the DP problem is decomposed into \((\{ 15 \}, \mathcal{R}, \mu, i)\) (which is easily solved by the reduction pair processor) and \((\{ 11 \}, 20), (13), \mathcal{R}, \mu, i)\).

Now we apply the reduction pair processor again with the following rational polynomial interpretation [11]: \( [M(x, y)] = \frac{1}{3}x + \frac{1}{2}y, [\text{minus}(x, y)] = 2x + \frac{1}{2}y, [\text{gt}(x, y, z)] = \frac{1}{3}x + y + z, [\text{gt}(x, y, z)] = \frac{1}{3}x + y + z, [\text{U}(x)] = x, [p(x)] = g(x), [s(x)] = 2x + 2, [\text{true}] = 1, [\text{false}] = 0 \). Then (20) is strictly decreasing and can be removed, whereas all other remaining DPs and usable rules

15 In the papers on CS-DPs up to now, the only existing adaption of such a processor was the straightforward adaption of the narrowing processor in the case \( e = t \), cf. [2]. However, this processor would not help for the TRS of Ex. 1.

16 The counterexample of [4, Ex. 8] in Footnote 11 again illustrates why \( \text{Ren}^\nu \) is also needed in the innermost case (whereas this is unnecessary for non-CS rewriting).
are weakly decreasing. A last application of the dependency graph processor then
detects that there is no cycle anymore and thus, it returns the empty set of DP
problems. Hence, termination of the TRS from Ex. 1 is proved. As shown in our
experiments in Sect. 5, this proof can easily be performed automatically.

5 Experiments and Conclusion

We have developed a new notion of context-sensitive dependency pairs which
improves significantly over previous notions. There are two main advantages:

(1) **Easier adaption of termination techniques to CS rewriting**
Now CS-DPs are very similar to DPs for ordinary rewriting and consequently,
the existing powerful termination techniques from the DP framework can
easily be adapted to context-sensitive rewriting. We have demonstrated this
with some of the most popular DP processors in Sect. 4. Our adaptions
subsume the existing earlier adaptions of the dependency graph [2], of the
usable rules [20], and of the modifications for innermost rewriting [4], which
were previously developed for the notion of CS-DPs from [1].

(2) **More powerful termination analysis for CS rewriting**
Due to the absence of collapsing CS-DPs, one does not have to impose extra
restrictions anymore when extending the DP processors to CS rewriting, cf.
Ex. 23. Hence, the power of termination proving is increased substantially.

To substantiate Claim (2), we performed extensive experiments. We imple-
mented our new non-collapsing CS-DPs and all DP processors from this paper
in the termination prover AProVE [16]. 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 termina-
tion of 78 examples (including all 68 TRSs where MU-TERM is successful). Since 4 examples are known to be non-terminating, at most 8 more of the 90
examples could potentially be detected as terminating. So due to the results of
this paper, termination proving of context-sensitive rewriting has now become

---

17 We also used the subterm criterion and forward instantiation processors, cf. Sect. 4.
18 If AProVE is restricted to use exactly the same processors as MU-TERM, then it still
succeeds on 74 examples. So its superiority is indeed mainly due to the new CS-DPs
which enable an easy adaption of the DP framework to the CS setting.
very powerful. To experiment with our implementation and for details, we refer to http://aprove.informatik.rwth-aachen.de/eval/CS-DPs/.

References

8.11 Context-Sensitive Dependency Pairs

Context-sensitive dependency pairs

Beatriz Alarcón, Raúl Gutiérrez, Salvador Lucas

ELP Group, DSIC, Universidad Politécnica de Valencia, Spain

ARTICLE INFO

Article history:
Received 31 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
first imposed on the arguments of function symbols \( f \) in the signature \( \mathcal{F} \). A signature is a set of function symbols \( f_1, \ldots, f_k \) together with an arity function \( \text{ar}: \mathcal{F} \to \mathbb{N} \) that establishes the number of ‘arguments’ associated to each symbol. A replacement map is a mapping \( \mu: \mathcal{F} \to \mathcal{P}(\mathbb{N}) \) that satisfies \( \mu(f) \subseteq \{1, \ldots, \text{ar}(f)\} \), 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 \( f(t_1, \ldots, t_k) \) if \( i \in \mu(f) \).

Example 1. The TRS \( R \) in Fig. 1 can be used to compute approximations to \( \frac{\pi}{2} \) by using Wallis’ product: \( \frac{\pi}{2} = \lim_{n \to \infty} \frac{2n!}{(n!)^2} \)...

208 Publications (full text)
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 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 \( g \). 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 \( u = f(l_1, \ldots, l_k) \) if \( l = f(l_1, \ldots, l_k) \), and where \( v = g^s(l_1, \ldots, l_m) \) if \( s = g(l_1, \ldots, l_m) \) is a subterm of \( r \) and \( g \) is a defined symbol. The notation \( f^s \) for a given symbol \( f \) means that \( f \) is marked. In

---

*Fig. 2. Dependency pairs for the TRS in Example 1.*
practice, we often capitalize f and use F instead of f^2 in our examples. For this reason, the dependency pair technique starts by considering a new TRS DP(\mathcal{R}) that contains all these new rules for each I \rightarrow r \in \mathcal{R}. For instance, according to [10], the set 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 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 \mu_i \rightarrow \nu_i of dependency pairs together with a substitution \sigma such that \sigma(\nu_i) rewrites to \sigma(\mu_{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 I \rightarrow r which are considered to build the context-sensitive dependency pairs F \rightarrow F must be \mu-replacing terms now.

Example 2. Consider \mathcal{R} and \mu as in Example 1. Only the dependency pairs (1), (4)\rightarrow(12), (14)\rightarrow(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}:

\begin{align*}
a \rightarrow c(f(a)) \\
F(c(x)) \rightarrow x
\end{align*}

together with \mu(c) = \emptyset and \mu(f) = \{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 \lambda \rightarrow \lambda 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:

\begin{align*}
F(c(x)) & \quad \mu \\
F(c(f(a))) & \quad \mu \\
F(f(a)) & \quad \mu \\
\cdots
\end{align*}

Indeed, we must add the following collapsing dependency pair:

\begin{align*}
F(c(x)) \rightarrow x.
\end{align*}

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 I \rightarrow r which is used in the sequence. Migrating variables are those that are not replacing in the left-hand side I 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:

\begin{align*}
\text{T\!A\!I\!L}(\text{cons}(x, xs)) & \rightarrow xs \\
\text{T\!A\!K\!E}(\text{cons}(x, xs)) & \rightarrow x \\
\text{Note that variable xs is } & \mu\text{-replacing in the right-hand sides of the rules } & \text{T\!A\!I\!L}(\text{cons}(x, xs)) & \rightarrow xs \text{ and } & \text{T\!A\!K\!E}(\text{cons}(x, xs)) & \rightarrow \text{cons}(x, \text{T\!A\!K\!E}(\text{cons}(x, xs))) & \text{but it is non-} \mu\text{-replacing in the corresponding left-hand sides.}
\end{align*}
1.3. Plan of the paper

We have argued that termination of CSR is an interesting and challenging topic of research with a good number of practical applications. The results, techniques, and tools that derive from our work can be of interest to a sufficiently wide audience.

The material in this paper will be more familiar, however, to those specialists who are interested in termination (in general) and in how to prove termination of CSR in particular. Throughout the paper, however, we made a serious effort to provide sufficient intuition and informal descriptions for our main definitions and results.

After Section 2, the paper is structured in three main parts:

1. Section 3 provides appropriate notions of minimal non-μ-terminating terms and introduces the main properties of such terms. We introduce the notion of hidden term and investigate the structure of infinite context-sensitive rewrite sequences starting from minimal non-μ-terminating terms. This analysis is essential in order to provide an appropriate definition of context-sensitive dependency pair and the related notions of chains, graphs, etc.

2. We define the notions of context-sensitive dependency pair and context-sensitive chain of pairs and show how to use them to characterize termination of CSR. Sections 4 and 5 introduce the general framework to compute and use context-sensitive dependency pairs to prove termination of CSR. The introduction of dependency pairs of a new kind (the collapsing dependency pairs, as in Example 3) leads to a notion of context-sensitive dependency chain, which is quite different from the standard one. In Section 6, we prove that our context-sensitive dependency pair approach fully characterizes termination of CSR.

3. We describe a suitable framework for dealing with proofs of termination of CSR by using these results. Section 7 adapts the dependency pair framework [33,35] to CSR by defining appropriate notions of CS problem and CS processor that rely on the results obtained in the second part of the paper. Section 8 introduces the notion of context-sensitive (dependency) graph and the associated CS processor. Section 9 describes CS processors for removing or transforming collapsing pairs. Section 10 investigates the use of term orderings in processors. Section 11 adapts Hirokawa and Middeldorp’s subterm criterion [38]. Section 12 adapts narrowing transformation of pairs in [35].

Experiments are reported in Section 13. Sections 14 and 15 discuss related work. Section 16 concludes.
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 μ such that μ(x) ∈ X for all x ∈ X. A substitution σ such that σ(s) = σ(t) for two terms s, t ∈ T(F, X) is called a unifier of s and t; we also say that s and t unify (with substitution σ). If two terms s and t unify, then there is a unique most general unifier σ (up to renaming of variables) such that for every other unifier τ, there is a substitution θ such that θ ◦ σ = τ.

A relation R ⊆ T(F, X) × T(F, X) on terms is stable if, for all terms s, t ∈ T(F, X) and substitutions σ, we have σ(s)Rσ(t) whenever sRt.

2.3. Rewrite systems and term rewriting

A rewrite rule is an ordered pair (l, r), written l → r, with l, r ∈ T(F, X) and Var(r) ⊆ Var(l). The left-hand side (lhs) of the rule is l, and the right-hand side (rhs) is r. A rewrite rule l → r is said to be collapsing if r ∈ Y. A Term Rewriting System (TRS) is a pair R = (F, R), where K is a set of rewrite rules. We often use ⊢ to denote TRSs whose set of rules R is empty. Given TRSs R = (F, R) and R′ = (F, R′), we let R ∪ R′ be the TRS (F, R ∪ R′). An instance σ(t) of a rhs of a rule is called a redex. Given R = (F, R), we consider F as the disjoint union F = C ∪ D of symbols c ∈ C (called constructors) and symbols f ∈ D (called defined functions), where D = {root(l) | l → r ∈ R} and C = F − D.

Example 5. Consider again the TRS in Example 1. The symbols even, odd, incr, take, zip, tail, rep3, add, prod, prodFrac, prodOffFrac, and halfFrac are defined. Symbols a, b, cons, consF, nil, and frac are constructors.

We often write l → r ∈ R instead of l → r ∈ R to express that the rule l → r is a rule of R. A term s ∈ T(F, X) rewrites to t (at position p), written s →p R t (or just s → R t, or s → t), if s|p = σ(l) and t = sσ(r)|p, for some rule l → r ∈ R, p ∈ Par(s) and substitution σ. We write s →n R t for some n ≥ p. A TRS R is terminating if its one step rewrite relation →R is terminating.

2.4. Context-sensitive rewriting

A mapping μ : F → ω(Ω) is a replacement map (or F-map) if for all symbols f ∈ F, μ(f) ⊆ {1, ..., ar(f)} [44]. Let M be the set of all F-maps (or M for the F-maps of a TRS (F, R)). Let μ be the replacement map given by μ(f) = {1, ar(f)} for all f ∈ F (i.e., no replacement restrictions are specified).

A binary relation R on terms is μ-monotonic if, for all f ∈ F, l, m ∈ μ(f), and s, t, t1, ..., tn ∈ T(F, X), f(t1, ..., tn), s, l, m, l1, ..., ln, M whenever sRt. If R is μ-monotonic, we just say that R is monotonic.

The set of μ-replacing positions Posμ(t) of t ∈ T(F, X) is: Posμ(t) = {A} if t ∈ X, and Posμ(t) = {A} ∪ ∪i∈Par(t) μ(i)Posμ(t[i]) if t /∈ X. Note that Posμ(t)(as Posμ(t)) is prefix closed. When no replacement map is made explicit, the μ-replacing positions are often called active; and the non-μ-replacing ones are often called frozen. The following results about CR are often used without any explicit mention.

Proposition 1 [44]. Let t ∈ T(F, X) and p = q · q′ ∈ Posμ(t). Then p ∈ Posμ(t) if and only if q ∈ Posμ(t) ∧ q′ ∈ Posμ(t[q]).

The chain of symbols lying on positions above/on p ∈ Posμ(t) is prefix(A) = root(t), prefix(p · i) = root(t), prefix(p). The strict prefix s is prefixf(A) = A, prefixf(p · i) = prefixf(p), i.e., the last symbol in prefixf(p · i) is removed. Although prefixf(p) is a sequence, when the ordering of symbols in prefixf(p) does not matter, we often use the standard set-theoretic notation (e.g., inclusion as in prefixf(p) ⊆ F) with the obvious meaning.

Proposition 2 [44]. If p ∈ Posμ(t) ∩ Posμ(s) and prefixf(p) = prefixf(s), then p ∈ Posμ(t) ∧ p ∈ Posμ(s).

The μ-replacing subterm relation ⊆μ is given by s ⊆μ t if there is p ∈ Posμ(s) such that t = s|p. We write s ⊆μ t if s ⊆μ t and s /∈ t. We write s ⊆μ δ t to denote that t is a non-μ-replacing (hence strict) subterm of s; s ⊆μ δ t if there is p ∈ Posμ(s) − Posμ(s) such that t = s|p. The set of μ-replacing positions of a term t, i.e., variables occurring at some non-μ-replacing position in t, is Varμ(t) = {x ∈ Var(t) | t ⊆μ t[x]}. Note that Varμ(t) and Varδ(t) do not need to be disjoint (when t is not linear).
A pair \((R, \mu)\) where \(R\) is a TRS and \(\mu \in M_R\) is often called a CS-TRS. In context-sensitive rewriting, we (only) contract \(\mu\)-replacing redexes: \(s \rightarrow_{\mu} t\) rewritten \(s \leftarrow_{\mu} t\) (or \(s \rightarrow_{\mu} t\), \(s \leftarrow_{\mu} t\) and even \(s \leftarrow_{\mu} t\), if \(s \rightarrow_{\mu} t\) and \(p \in \text{Pos}^\mu(s)\).

**Example 6.** Consider \(R\) and \(\mu\) as in Example 1. Then, we have:

\[ \text{odd}_s \rightarrow_{\mu} \text{cons}(0, \text{incr}(\text{odd}_s)) \]

Since the second argument of \(\text{cons}\) is not \(\mu\)-replacing, we have \(2 \not\in \text{Pos}^\mu(\text{cons}(0, \text{incr}(\text{odd}_s)))\). Thus, redex \(\text{odd}_s\) cannot be \(\mu\)-rewritten.

A term \(t\) is \(\mu\)-terminating (or \((R, \mu)\)-terminating, if we want an explicit reference to the involved TRS \(R\)) if there is no infinite \(\mu\)-rewrite sequence \(t = t_1 \rightarrow_{\mu} t_2 \rightarrow_{\mu} \cdots \rightarrow_{\mu} t_n \rightarrow_{\mu} \cdots \) starting from \(t\). A TRS \(R\) is \(\mu\)-terminating if \(\rightarrow_{\mu}\) is terminating.

A term \(s\) \(\mu\)-narrows to a term \(t\) (written \(s \sim_{\mu} t\)), if there is a nonvariable \(\mu\)-replacing position \(p \in \text{Pos}^\mu(s)\) and a rule \(l \rightarrow r \in R\) (sharing no variable with \(s\)) such that \(s|_p\) and \(t|_l\) unify with the most general unifier \(\theta\) and \(t = \theta(s|_p)\). 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 \(\hat{\text{mb}}^\mu(\mathcal{F})\) consists of the following rules:

\[ [f(x_1, \ldots, x_n) \rightarrow x_i \mid f \in \mathcal{F}, i \in \mu(f)] \]

### 3. Minimal non-\(\mu\)-terminating terms and infinite \(\mu\)-rewrite sequences

Given a TRS \(R = (\mathcal{C} \cup \mathcal{D}, R)\), the minimal nonterminating terms associated to \(R\) are nonterminating terms \(t\) whose proper subterms \(u\) (i.e., \(t \triangleright u\)) are terminating; \(T_{\text{nc}}\) is the set of minimal nonterminating terms associated to \(R\) [38,40]. Minimal nonterminating terms have two important properties:

1. Every nonterminating term \(s\) contains a minimal nonterminating term \(t \in T_{\text{nc}}\) (i.e., \(s \ni t\)).
2. Minimal nonterminating terms \(t\) are always rooted by a defined symbol \(f \in \mathcal{D} \cup \mathcal{C} \triangleright \mathcal{T}_{\text{nc}}\), root(t) \(\in \mathcal{D}\).

As discussed in [38], considering the structure of the infinite rewrite sequences starting from a minimal nonterminating term \(t \in T_{\text{nc}}\) can be helpful to come to the notion of dependency pair [10]. Such sequences proceed as follows:

**Proposition 3 [38, Lemma 1].** Let \(R = (\mathcal{C} \cup \mathcal{D}, R)\) be a TRS. For all \(t \in T_{\text{nc}}\), there exist \(l \rightarrow r \in R\), a substitution \(\sigma\) and a term \(u \in T_{\text{nc}}\) such that root\((u)\) \(\in \mathcal{D}\), \(l \rightarrow \sigma\left(l\right) \rightarrow \sigma\left(r\right) \triangleright u\), and there is a nonvariable subterm \(v\) of \(r\) such that \(u \triangleright v\), such that \(u = \sigma\left(v\right)\).

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.\(^7\)

**Lemma 1.** Let \(R = (\mathcal{F}, \mu)\) be a TRS, \(\mu \in M_{\mathcal{F}}\), and \(s, t \in T(\mathcal{F}, X)\). If \(s\) is \(\mu\)-terminating, then:

1. If \(s \ni t\), then \(t\) is \(\mu\)-terminating.
2. If \(s \rightarrow_{\mu} t\), then \(t\) is \(\mu\)-terminating.

Given a TRS \(R = (\mathcal{F}, \mu)\) 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., \(T_{\text{nc}}\)), is the following: let \(T_{\text{nc}, \mu}\) be a set of minimal non-\(\mu\)-terminating terms in the following sense: \(t\) belongs to \(T_{\text{nc}, \mu}\) if \(t\) is non-\(\mu\)-terminating and every strict subterm \(u\) (i.e., \(t \triangleright u\)) is \(\mu\)-terminating. It is obvious that root\((t)\) \(\in \mathcal{D}\) for all \(t \in T_{\text{nc}, \mu}\). We also have the following:

**Lemma 2.** Let \(R = (\mathcal{F}, \mu)\) be a TRS, \(\mu \in M_{\mathcal{F}}\), and \(s \in T(\mathcal{F}, X)\). If \(s\) is not \(\mu\)-terminating, then there is a subterm \(t\) of \(s\) such that \(t \in T_{\text{nc}, \mu}\).

\(^7\) 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 $T_{M,\mu}$.

**Example 7.** Consider the CS-TRS $(R, \mu)$ in Example 3 and $s = \varepsilon(c(t(a)))$. Note that $s$ is not $\mu$-terminating, but $s \notin T_{M,\mu}$ because $\varepsilon(c(t(a))) \triangleright_\mu \varepsilon(t(a))$ and $\varepsilon(t(a))$ is not $\mu$-terminating. Note that $\varepsilon(c(t(a))) \triangleright_\mu \varepsilon(t(a))$. The only $\mu$-replacing strict subterm of $s$ is $\varepsilon(t(a))$, which is $\mu$-terminating, i.e., $\varepsilon(t(a)) \notin T_{M,\mu}$.

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 $M_{\mu}$ be a set of minimal non-$\mu$-terminating terms in the following sense: $t$ belongs to $M_{\mu}$ 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 $T_{M,\mu} \subseteq M_{\mu}$. In the following, we often say that terms in $T_{M,\mu}$ are strongly minimal non-$\mu$-terminating; we use them in Section 3.4. Now, we have the following:

**Lemma 3.** Let $R = (F, R)$ be a TRS, $\mu \in M_F$, and $s \in T(F, X)$. If $s$ is not $\mu$-terminating, then there is a $\mu$-replacing subterm $t$ of $s$ such that $t \in T_{M,\mu}$.

Obviously, if $t \in M_{\mu}$, then root$(t)$ is a defined symbol. Since $\mu$-terminating terms are preserved under $\mu$-rewriting (Lemma 1), it follows that $M_{\mu}$ is preserved under inner $\mu$-rewritings in the following sense.

**Lemma 4.** Let $R$ be a TRS, $\mu \in M_F$, and $t \in M_{\mu}$. If $t \xrightarrow{\Delta}^* u$ and $u$ is non-$\mu$-terminating, then $u \in M_{\mu}$.

Lemma 4 does not hold for $T_{M,\mu}$; consider the CS-TRS $(R, \mu)$ in Example 3. Note that $\varepsilon(\varepsilon(t(a))) \in T_{M,\mu}$ and $\varepsilon(\varepsilon(t(a))) \xrightarrow{\Delta}^* \varepsilon(\varepsilon(t(a)))$. Although $\varepsilon(\varepsilon(t(a)))$ is not $\mu$-terminating, $\varepsilon(\varepsilon(\varepsilon(t(a)))) \notin T_{M,\mu}$, as shown in Example 7.

### 3.2. Hidden terms in minimal $\mu$-rewrite sequences

Given a CS-TRS $(R, \mu)$, the hidden terms are nonvariable terms occurring on some frozen position in the right-hand side of some rule of $R$. As we show in the next section, they play an important role in infinite minimal $\mu$-rewrite sequences associated to $R$.

**Definition 3** (Hidden symbols and terms). Let $R = (F, R)$ be a TRS and $\mu \in M_F$. We say that $t \in T(F, X) - X$ is a hidden term if there is a rule $l \rightarrow r \in R$ such that $t \triangleright_\mu l$. Let $H(T(R, \mu))$ (or just $H$ if no confusion arises) be the set of all hidden terms in $(R, \mu)$. We say that $f \in F$ is a hidden symbol if it occurs in a hidden term. Let $H(R, \mu)$ (or just $H$) be the set of all hidden symbols in $(R, \mu)$.

In the following, we also use $D(H(T(R, \mu)) = \{ l \in H(T(R, \mu) | \text{root}(l) \in D \}$ for the set of hidden terms which are rooted by a defined symbol.

**Example 8.** For $R$ and $\mu$ as in Example 1, the maximal hidden terms are $\text{incr}(\text{odd}(x))$, $\text{incr}(x)$, $\text{zip}(x, y)$, and $\text{con}(x, \text{zip}(x))$. The hidden symbols are $\text{odds}$, $\text{odds}$, $\text{incr}$, $\text{incr}$, $\text{incr}$, $\text{con}$, and $\text{con}$. Finally, $D(H(T(R, \mu)) = \{ \text{odds} \text{, incr(odd(x))} \text{, incr}(x) \text{, zip}(x, y) \text{, zip}(x, y) \text{, zip}(x, y) \}$. The following lemma says that frozen subterms $t$ in the contractum $\sigma(r)$ of a redex $\sigma(l)$ that do not contain $r$ 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 $R = (F, R)$ be a TRS and $\mu \in M_F$. Let $l \in T(F, X)$ 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 $t' \in H$ such that $r \triangleright_\mu r'$ and $\sigma(r') \triangleright_\mu r$.

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 \( R \) be a TRS and \( \mu \in M_{L} \). Let \( A \) be a \( \mu \)-rewrite sequence \( t_1 \rightarrow \mu t_2 \rightarrow \mu \cdots \rightarrow \mu t_n \) with \( t_i \in M_{\mu,C} \) for all \( i \), \( 1 \leq i \leq n \). If there is a term \( t \in M_{\mu,C} \) such that \( t_1 \not\rightarrow \mu t \) and \( t_2 \not\rightarrow \mu t \), then \( t = \sigma(s) \) for some \( s \in \text{NTT} \) 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 \( R \) be a TRS and \( \mu \in M_{L} \). Consider a finite or infinite sequence of the form \( t_1 \rightarrow \mu t_2 \rightarrow \mu \cdots \rightarrow \mu t_n \) with \( t_i \in M_{\mu,C} \) for all \( i \geq 1 \). If there is a term \( t \in M_{\mu,C} \) such that \( t \not\rightarrow \mu t \) for some \( i \geq 1 \), then \( t_1 \not\rightarrow \mu t_2 \not\rightarrow \mu t \) for some \( s \in \text{NTT} \) 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 \( t \in M_{\mu,C} \), 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 \( x \in \text{Var}(r) - \text{Var}(l) \) that are not \( \mu \)-replacing in the left-hand side but become \( \mu \)-replacing in the right-hand side.

Proposition 5. Let \( R \) be a TRS and \( \mu \in M_{L} \). Then, for all \( t \in M_{\mu,C} \), there exist \( l \not\rightarrow r \in R \), a substitution \( \sigma \), and a term \( u \in M_{\mu,C} \) such that \( t \not\rightarrow \sigma(l) \not\rightarrow \sigma(r) \not\rightarrow \mu u \) and either

1. there is a nonvariable \( \mu \)-replacing subterm \( s \) of \( l \) such that \( \tau(s) = \mu(s) \), or
2. there is \( x \in \text{Var}(r) - \text{Var}(l) \) such that \( \sigma(x) \not\in R \).

Proof. Consider an infinite \( \mu \)-rewrite sequence starting from \( t \). By definition of \( M_{\mu,C} \), 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 \( l \not\rightarrow r \in R \):

\[ t \not\rightarrow \sigma(l) \not\rightarrow \sigma(r) \not\rightarrow \mu \tau \]

Thus, we can write \( t = f(t_1, \ldots, t_k) \) and \( \sigma(l) = f(l_1, \ldots, l_k) \) for some \( k \)-ary defined symbol \( f \), and \( \sigma(l_i) \not\rightarrow \mu \tau \) for all \( 1 \leq i \leq k \). Since all \( l_i \) are \( \mu \)-terminating for \( t \), 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 \( l \) such that \( \tau(y) \not\in \text{Var}(l) \). Since \( \sigma(y) \) is \( \mu \)-terminating, by Lemma 3, it contains a \( \mu \)-replacing subterm \( u \in M_{\mu,C} \) such that \( \sigma(y) \not\in \text{Var}(u) \), i.e., there is a position \( p \in \text{Pos}(\sigma(y)) \) such that \( \sigma(y)_p = u \). We consider two cases:

1. If \( p \in \text{Pos}(\tau(y)) \), then there is a nonvariable position \( r \) of \( y \) in \( \tau(y) - \text{Var}(r) \) such that \( \tau(y)_r = \sigma(y)_r \), and subterm \( u \) is \( \mu \)-replacing.
2. If \( p \not\in \text{Pos}(\tau(y)) \), then there is a \( \mu \)-replacing variable \( q \in \text{Var}(\tau(y)) \cap \text{Var}(\tau(y)) \) such that \( q \not\in p \). Let \( x \in \text{Var}(\tau(y)) \) be such that \( \tau(y)_x = q \).

Thus, \( \sigma(x)_p = \tau(y)_x \), and \( \sigma(x) \) is not \( \mu \)-terminating since \( u \in M_{\mu,C} \) is not \( \mu \)-terminating, by Lemma 1. \( \sigma(x) \) is not \( \mu \)-terminating. Since \( \tau(y) \) is \( \mu \)-terminating for all \( y \in \text{Var}(\tau(y)) \), we conclude that \( x \in \text{Var}(\tau(y)) - \text{Var}(\tau(y)) \).

Proposition 5 entails the following result, which establishes some properties of infinite sequences starting from minimal non-\( \mu \)-terminating terms.

Corollary 1. Let \( R \) be a TRS and \( \mu \in M_{L} \). For all \( t \in M_{\mu,C} \), there is an infinite sequence

\[ t \not\rightarrow \sigma(t_1) \not\rightarrow \sigma(t_2) \not\rightarrow \mu \cdots \not\rightarrow \mu \tau \]

where, for all \( i \geq 1 \), \( t_i \not\rightarrow r \in R \) are rewrite rules, \( \sigma \) are substitutions, and terms \( t_i \in M_{\mu,C} \) are minimal non-\( \mu \)-terminating terms such that either

1. \( t_i = \sigma(t) \) for some nonvariable subterm \( t \) such that \( t \not\rightarrow \mu t \) or
2. \( \sigma(x) \not\in \text{Var}(\tau(t)) \) for some \( x \in \text{Var}(\tau(t)) - \text{Var}(t) \).

Remark 2. The \( \leftrightarrow \mu \not\rightarrow \mu \) sequence in Corollary 1 can be easily viewed as an infinite \( \mu \)-rewrite sequence by just introducing appropriate contexts \( C[l, t_i] \) with \( \mu \)-replacing holes: since \( \sigma(t_i) \not\rightarrow \mu t_i \), there is \( p_i \in \text{Pos}(\sigma(t)) \) such that \( \sigma(t_i) = \sigma(t_i)[p_i] \); just take \( C[l] = \sigma(t)[p] \). Hence:
\[ t \overset{*}{\rightsquigarrow} \sigma(t_1) \rightsquigarrow C_1[t_1]_{\sigma} \overset{*}{\rightsquigarrow} C_2[t_2]_{\sigma} \rightsquigarrow \cdots \]

Note that, e.g., \( t_1 \vdash t_2 \) in \( \tau(t) \leftrightarrow \rho(t) \) (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 \( \forall \varphi(t) \):

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

Note that \( \text{REN}^\mu(t) \) keeps variables at non-\( \mu \)-rewriting positions untouched. Note also that \( \text{REN}^\mu \) is a substitution: it replaces the \( n(s) \) different \( \mu \)-rewriting occurrences of the same variable \( x \) by different variables \( x_1, \ldots, x_n(s) \). 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 \( 1 \leq i \leq n(s) \)). The use of \( \text{REN}^\mu(t) \) together with \( \mu \)-narrowability yields a necessary condition for reducibility of terms under some instantiations which is used in our development.

**Proposition 6**. Let \( \mathcal{R} = (F, \mathcal{R}) \) be a TRS and \( \mu \in \mathcal{M}_F \). Let \( t \in \mathcal{T}(F, \mathcal{X}) - \mathcal{X} \) be a nonvariable term and \( \sigma \) be a substitution. If \( \sigma(t) \overset{\Delta}{\rightarrow} \sigma(l) \) 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: \( \sigma(t) = t_1 \overset{\Delta}{\rightarrow} t_2 \overset{\Delta}{\rightarrow} \cdots \overset{\Delta}{\rightarrow} t_m = \sigma(l) \) for some \( m \geq 1 \). We proceed by induction on \( m \).

1. If \( m = 1 \), then \( \sigma(t) = \sigma(l) \). Since \( t \not\in \mathcal{X} \), \( t \) 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 \( \mu \)-grio. \( \mu \). Since \( t \not\in \mathcal{X} \), implies that \( \text{REN}^\mu(t) \not\in \mathcal{X} \). \( \text{REN}^\mu(t) \) is \( \mu \)-narrowable at the root position using the same rule \( l \rightarrow r \).

2. If \( m > 1 \), then we have \( t_1 \overset{\Delta}{\rightarrow} t_2 \overset{\Delta}{\rightarrow} \cdots \overset{\Delta}{\rightarrow} t_m = \mu(l) \). We consider two cases according to the position \( p \in \mathcal{T}(\mu(t)) \) where the \( \mu \)-rewriting step \( t_1 \overset{\Delta}{\rightarrow} t_2 \) is performed (note that \( t_1 = \mu(l) \) by assumption).

(a) If \( p \in \mathcal{T}(\mu(t)) \), then there is a rule \( l' \rightarrow r' \) and a substitution \( \theta \) such that \( \sigma(t)_{\theta} = \sigma(l') \). Again, we have \( \sigma(t)_{\theta} = \sigma(l') \). Thus, \( t \) is \( \mu \)-narrowable at position \( p \) using rule \( l' \rightarrow r' \). (Reasoning as above), we conclude that \( \text{REN}^\mu(t) \) is \( \mu \)-narrowable.

(b) If \( p \not\in \mathcal{T}(\mu(t)) \), then there is a \( \mu \)-rewriting variable position \( q \in \mathcal{T}(\mu(t)) \) of \( t \) such that \( l_q = x \in \forall \varphi(t), q \leq p \) and \( x \) is not \( \mu \)-grio. Therefore, \( t_1 = \sigma(t)_{\theta,x} = \sigma(t)_{\theta}(x) \) and \( t_2 = \sigma(t)_{\theta,x} = \sigma(t') \) for a term \( t' = t'_q \) where \( y \) is a new fresh variable \( y \not\in \forall \varphi(t) \) and a substitution \( \sigma' \) given by \( \sigma'(y) = t'_q \) and \( \sigma'(x) = \sigma(x) \) for all \( x \in \forall \varphi(t) \) (including \( x \)). Clearly, \( \sigma'(t') = \sigma'(t)'(x)' = \sigma(t)_{\theta}(x) = \sigma(t)_{\theta,x} = t_2 \).

By the induction hypothesis, \( \text{REN}^\mu(t') \) is \( \mu \)-narrowable. Since \( t' \) only differs 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.

**Corollary 2**. Let \( \mathcal{R} = (F, \mathcal{R}) \) be a TRS and \( \mu \in \mathcal{M}_F \). Let \( t \in \mathcal{T}(F, \mathcal{X}) - \mathcal{X} \) be a nonvariable term and \( \sigma \) be a substitution such that \( \sigma(t) \in \mathcal{M}_{\mu,\mu} \). 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 \( \sigma(t) \overset{\Delta}{\rightarrow} \sigma(l) \) (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.

In the following, we write \( \text{NAR}^\mu(t) \) (or just \( \text{NAR}^\mu(t) \)) to indicate that \( t \) is \( \mu \)-narrowable with respect to the (intended) TRS \( \mathcal{R} \). We also let

\[ \text{NAR}^\mu(t, \mu) = \{ t \in \forall \varphi(N, \mu) | \text{NAR}^\mu(t) \} \]

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 \( t \in DNT(R, \mu) \) for \( R \) and \( \mu \) as in Example 8 are \( \mu \)-narrowable (even without applying \( \text{Ren}^0 \)), we have \( NHT(R, \mu) = DNT(R, \mu) \).

As a consequence of the previous results, we have the following main result, which we use later.

Theorem 1. Let \( R \) be a TRS and \( \mu \in M_R \). For all \( t \in T_{\infty, \mu} \), there is an infinite sequence

\[
t = t_0 \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdots
\]

where, for all \( i \geq 1, t_i \rightarrow R \) are rewrite rules, \( \sigma_i \) are substitutions, and terms \( t_i \in M_{\infty, \mu} \) are minimal non-\( \mu \)-terminating terms such that either

1. \( \sigma_i(s_i) \) for some nonvariable term \( s_i \) such that \( t_i \overset{\mu}{\Rightarrow} s_i \), or
2. \( \sigma_i(s_i) \overset{\mu}{\Rightarrow} t_i \) for some nonvariable term \( s_i \in \text{Var}^R(t_i) = \text{Var}^R(t_i) \) and \( t_i = \theta(t'_i) \) for some \( t'_i \in NHT \) and substitution \( \theta \).

Proof. Since \( T_{\infty, \mu} \subseteq M_{\infty, \mu} \), by Corollary 1, we have a sequence

\[
t = t_0 \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \overset{\lambda}{\longrightarrow} \cdot \sigma_2(t_2) \overset{\lambda}{\longrightarrow} \cdots
\]

where, for all \( i \geq 1, t_i \rightarrow R \), \( \sigma_i \) are substitutions, \( t_i \in M_{\infty, \mu} \), and either (1) \( t_i = \sigma_i(s_i) \) for some nonvariable term \( s_i \) such that \( t_i \overset{\mu}{\Rightarrow} s_i \) or (2) \( \sigma_i(s_i) \overset{\mu}{\Rightarrow} t_i \) for some \( s_i \in \text{Var}^R(t_i) = \text{Var}^R(t_i) \) (and hence \( \sigma_i(t_i) \overset{\mu}{\Rightarrow} t_i \) and \( \sigma_i(t_i) \overset{\mu}{\Rightarrow} t_i \) as well). We only need to prove that terms \( t_i \) are instances of hidden terms in \( NHT \) whenever (2) holds. By Proposition 4, for all such terms \( t_i \), we have that either (A) \( \sigma_i(t_i) \overset{\mu}{\Rightarrow} t_i \) or (B) \( t_i = \theta(t'_i) \) for some \( t'_i \in DNT \) and substitution \( \theta \). In case (B), we just consider Corollary 2, which ensures that \( t'_i \in NHT \). In case (A), since \( t \overset{\lambda}{\longrightarrow} \cdot \sigma_1(t_1) \) and \( \sigma_1(t_1) \) is not \( \mu \)-terminating, by Lemma 4, all terms \( u_j \) in the \( \mu \)-rewrite sequence

\[
t = u_1 \overset{\lambda}{\longrightarrow} u_2 \overset{\lambda}{\longrightarrow} \cdots \overset{\lambda}{\longrightarrow} u_m = \sigma_1(t_1)
\]

belong to \( M_{\infty, \mu} \); \( u_j \in M_{\infty, \mu} \) for all \( j, 1 \leq j \leq m \). Since \( t \in T_{\infty, \mu} \), all its strict subterms (disregarding their \( \mu \)-replacing character) are \( \mu \)-terminating. Since \( t_i \) is not \( \mu \)-terminating, \( t \not\Rightarrow t_i \). By Lemma 6, \( t_i = \theta(t'_i) \) for some \( t'_i \in DNT \) and substitution \( \theta \). By Corollary 2, \( t'_i \in NHT \). \( \square \)

4. Context-sensitive dependency pairs

By Lemma 2 every non-\( \mu \)-terminating term \( t_0 \) contains a strongly minimal subterm \( t \in T_{\infty, \mu} \) 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 \( x \) which is a nonvariable \( \mu \)-replacing subterm of \( t \). By Corollary 2, \( \text{Ren}^0(x) \) 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 \( u \in M_{\infty, \mu} \), is an instance of a hidden term \( u' \in NHT \).

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 \( \bar{f} \) be a new fresh symbol (often called tuple symbol or DP-symbol) associated to a symbol \( f \). Let \( \mathcal{F}' \) be the set of tuple symbols associated to symbols in \( \mathcal{F} \). As usual, for \( t = f(t_1, \ldots, t_n) \in T(\mathcal{F}, \mathcal{X}) \), we write \( t^\mathcal{F} \) to denote the marked term \( t^\mathcal{F} = \bar{f}(t_1, \ldots, t_n) \). Conversely, given a marked term \( t^\mathcal{F} = \bar{f}(t_1, \ldots, t_n) \), where \( t_1, \ldots, t_n \in T(\mathcal{F}, \mathcal{X}) \), we write \( t^\mathcal{F} \) to denote the term \( f(t_1, \ldots, t_n) \in T(\mathcal{F}, \mathcal{X}) \).

Definition 4 (Context-sensitive dependency pairs). Let \( \mathcal{R} = (\mathcal{F}, \mathcal{X}) = (\mathcal{C} \uplus \mathcal{P}, \mathcal{R}) \) be a TRS and \( \mu \in M_\mathcal{F} \). Let \( \text{DP}_{\mathcal{F}}(R, \mu) = \text{DP}_{\mathcal{F}}(R, \mu) \cup \text{DP}_{\mathcal{F}}(R, \mu) \) be the set of context-sensitive dependency pairs (CSDPs) where:
We extend $\mu \in M_F$ into $\mu^\# \in M_F \cup D^\#$ by $\mu^\#(f) = \mu(f)$ if $f \in F$, and $\mu^\#(f^\#) = \mu(f)$ if $f \in D$.

The CSDPs $l \rightarrow v \in \text{DP}_F(R, \mu)$ 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 $l^\# \rightarrow s^\#$ are not subterms of the left-hand side (i.e., $\forall_{\mu} s$).
2. As in [53], we require $\mu$-narrowability of $\text{Ren}^\#(s)$: $\text{Narr}^\#(\text{Ren}^\#(s))$.

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 $R$ is $\mu$-conservative if $\text{Var}^\#(r) \subseteq \text{Var}^\#(l)$, i.e., there is no migrating variable; $R$ is $\mu$-conservative if all its rules are $\mu$-conservative (see [43,50]). The following fact is obvious from Definition 4.

Proposition 7. If $R$ is a $\mu$-conservative TRS, then $\text{DP}(R, \mu) = \text{DP}_F(R, \mu)$.

Therefore, in order to deal with $\mu$-conservative TRSs $R$ we only need to consider the ‘classical’ dependency pairs in $\text{DP}_F(R, \mu)$.

$$\text{ADD}(n, m) \rightarrow \text{ADD}(n, m) \tag{1}$$
$$\text{HALFP}(n) \rightarrow \text{EVENNS} \tag{4}$$
$$\text{HALFP}(n) \rightarrow \text{ODDNS} \tag{5}$$
$$\text{HALFP}(n) \rightarrow \text{PRODFRAC}\text{Fr}(n, z) \tag{6}$$
$$\text{HALFP}(n) \rightarrow \text{RTP}(z) \tag{7}$$
$$\text{HALFP}(n) \rightarrow \text{RTPI}(z) \tag{8}$$
$$\text{HALFP}(n) \rightarrow \text{TAIL}(z) \tag{9}$$
$$\text{HALFP}(n) \rightarrow \text{TAIL}(z) \tag{10}$$
$$\text{HALFP}(n) \rightarrow \text{TAXKE}(n, z) \tag{11}$$
$$\text{HALFP}(n) \rightarrow \text{ZIP}(z) \tag{12}$$
$$\text{ODDIS} \rightarrow \text{EVENNS} \tag{14}$$
$$\text{ODDIS} \rightarrow \text{ODDIS} \tag{15}$$
$$\text{PRODFRAC}(\frac{f}{g}, x) \rightarrow \text{PRODFRAC}(\frac{f}{g}, x) \tag{18}$$
$$\text{PRODFRAC}(\frac{f}{g}, x) \rightarrow \text{PRODFRAC}(\frac{f}{g}, x) \tag{19}$$
$$\text{PRODFRAC}(\text{const}(x, y), x) \rightarrow \text{PRODFRAC}(\text{const}(x, y), x) \tag{20}$$
$$\text{PRODFRAC}(\text{const}(x, y), x) \rightarrow \text{PRODFRAC}(\text{const}(x, y), x) \tag{21}$$
$$\text{TAXKE}(x, z) \rightarrow \text{TAXKE}(x, z) \tag{23}$$
$$\text{TAXKE}(x, z) \rightarrow \text{TAXKE}(x, z) \tag{24}$$
$$\text{TAXKE}(x, z) \rightarrow \text{TAXKE}(x, z) \tag{26}$$

Fig. 4. Context-sensitive dependency pairs for the CS-TRS in Example 1.
Example 10. Consider the following TRS $R$:

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

together with $\mu(g) = \mu(h) = \emptyset$ [63, Example 1]. Note that $R$ is $\mu$-conservative. DP$(R, \mu)$ consists of the following (non-collapsing) CSDPs:

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

with $\mu^1(c) = \mu^1(d) = \emptyset$.

If the TRS $R$ 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 $(R, \mu)$ 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 $R$ 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 $P$ together with $R$ 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 $P = DP(R, \mu)$.

Definition 5 (Chain of pairs - minimal chain). Let $R = (F, R)$ and $P = (G, P)$ be TRSs and $\mu \in M_{\mathcal{S}}$. A $(P, R, \mu)$-chain is a finite or infinite sequence of pairs $u_i \rightarrow v_i \in P$, together with a substitution $\sigma : X \rightarrow T(F \cup G, X)$ satisfying that, for all $i \geq 1$:

1. if $v_i \not\in \text{Var}(u_i) - \text{Var}(u_0)$, then $\sigma(v_i) \lessdot_{\mu, \mu} \sigma(u_{i+1})$, and
2. if $v_i \in \text{Var}(u_i) - \text{Var}(u_0)$, then $\sigma(v_i) = C_i[\bar{\sigma}]_h$ for some $C_i$ and $\bar{\sigma}$ such that $p_i \in P^{\text{csd}}(C_i[\bar{\sigma}])$, \text{sprefix}(C_i[\bar{\sigma}]) \subseteq F$, and $\imath_i[\bar{\sigma}] \lessdot_{\mu, \mu} \sigma(u_{i+1})$.

As usual, we assume that different occurrences of pairs do not share any variable (renaming substitutions are used if necessary). A $(P, R, \mu)$-chain is called minimal if for all $i \geq 1$:

1. if $v_i \not\in \text{Var}(u_i) - \text{Var}(u_0)$, then $\sigma(v_i) = (R, \mu)$-terminating, and
2. if $v_i \in \text{Var}(u_i) - \text{Var}(u_0)$, then $\imath_i[\bar{\sigma}] \lessdot_{\mu, \mu} \sigma(u_{i+1})$ and $\exists \bar{\sigma} \in \mathcal{S}T(R, \mu)$ such that $v_i = \sigma(\bar{\sigma})$.

Note that the condition $v_i \not\in \text{Var}(u_i) - \text{Var}(u_0)$ 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 $P$). The following conventions about the component $P = (G, P)$ of our chains will be observed during our development:

1. According to the usual terminology [35], we often call pairs the rules $u \rightarrow v \in P$.
2. We have to mark terms $u \in T(F, X)$ before connecting them to the instance $\sigma(u_{i+1})$ of the left-hand side of the next pair. Since marked symbols $\imath_f$ are fresh (with respect to the signature $F$ of the TRS $R$), we also assume that $P^{\text{csd}} \cap F = \emptyset$ and $F^{\text{csd}} \subseteq G$.
3. We assume that $P$ contains a finite set of rules. This is essential in many proofs.

In the following, the pairs in a CS-TRS $(P, \mu)$, where $P = (G, P)$, are partitioned according to their role in Definition 5 as follows:

$$P_\sigma = \{u \rightarrow v \in P \mid v \in \text{Var}(u) - \text{Var}(u_0)\} \quad \text{and} \quad P_\mu = P - P_\sigma$$

Remark 5 (Collapsing pairs). Note that all pairs in $P_\mu = (G, P_\mu)$ are collapsing. The rules in $P_\mu = (G, P_\mu)$ can be collapsing as well: a rewrite rule $f(x) \rightarrow x \in P$ with $\mu(f) = \{1\}$ does not belong to $P_\sigma$ but rather to $P_\mu$ because $x$ is not a migrating variable.
Despite this fact, we refer to \(P_X\) as the set of collapsing pairs in \(P\) because its intended role in Definition 5 is capturing the computational behavior of collapsing CSDPs in \(\hat{D}_P(X, \mu)\).

Remark 6. [Notation for chains]. In general, a \(\langle P, R, \mu \rangle\)-chain can be written as follows:

\[
\sigma(u_1) \xrightarrow{R, \mu} \circ \bigcirc \circ t_1 \xrightarrow{\theta, \mu} \sigma(u_2) \xrightarrow{R, \mu} \circ \bigcirc \circ t_2 \xrightarrow{\theta, \mu} \cdots
\]

where, for all \(i \geq 1\) and \(u_i \rightarrow v_i \in P\),

1. if \(u_i \rightarrow v_i \notin P_X\), then \(t_i = \sigma(v_i)\),
2. if \(u_i \rightarrow v_i \in P_X\), then \(t_i = s_i^j\) for some term \(s_i\) such that \(\sigma(v_i) = C_i[s_i]_{\mu}\) for some \(C_i \upharpoonright_{\mu}\) such that \(p_i \in P \circ \rho^n(C_i \upharpoonright_{\mu})\), and \(\rho\)-predecessor \(\upharpoonright (p) \subseteq P\).

This is denoted in a compact way by \(\sigma(u) \xrightarrow{R, \mu} \circ \bigcirc \circ t\), emphasizing that there is a \(\rho\)-step followed by either an equality step (as in (1)) or by \(\mu\)-replacing projection steps (restricted to symbols in \(F\)) 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 \(NHT_P(R, \mu) \subseteq NHT(R, \mu)\) (or just \(NHT_P\)) be as follows:

\[NHT_P(R, \mu) = \left\{ t \in NHT(R, \mu) \mid \exists u \rightarrow v \in P, \exists \delta, \theta(\delta) \xrightarrow{\bullet} \theta(u) \right\}\]

This set contains the narrowable hidden terms that ‘connect’ with pairs in \(P\).

Remark 7. Note that \(NHT_P\) is not computable, in general, due to the need for checking the reachability of \(\theta(\delta)\) from \(\theta(u)\) using CSR. Suitable overapproximations are discussed below (see Remark 10).

We let \(P_X\) denote the subTRS of \(P\) containing the rules whose migrating variables occur on non-\(\mu\)-replacing immediate subterms in the left-hand side:

\[P_X = \left\{ [f(u_1, \ldots, u_k) \rightarrow x \mid \exists ! i \leq k, i \neq f, x \in \upharpoonright \sigma(u_i)] \right\}\]

Proposition 8. Let \(R = (F, R)\) and \(P = (G, P)\) be TRSs and \(\mu \in M_{\rho_\cup}^\mu\).

1. If \(NHT_P = \emptyset\), then every infinite minimal \(\langle P, R, \mu \rangle\)-chain is an infinite minimal \(\langle P_X, R, \mu \rangle\)-chain and there is no infinite minimal \(\langle P_X, R, \mu \rangle\)-chain.
2. If \(P = P_X\), then there is no infinite \(\langle P, R, \mu \rangle\)-chain.

Proof.

1. By contradiction. Assume that there is an infinite minimal \(\langle P, R, \mu \rangle\)-chain containing any \(u_i \rightarrow v_i \in P_X\). By Definition 5, such a pair must be followed by a pair \(u_{i+1} \rightarrow v_{i+1} \in P\), such that \(\delta(C_i)^{s_i} \xrightarrow{\theta, \mu} \sigma(u_{i+1})\) for some \(s_i \in NHT\) and substitution \(\theta\). Therefore, \(t_i \in NHT_P\), but \(NHT_P = \emptyset\), leading to a contradiction.

2. By contradiction. Assume that there is an infinite chain that only uses dependency pairs \(u_i \rightarrow v_i \in P_X\) for all \(i \geq 1\). Let \(f_i = \text{root}(u_i)\) for \(i \geq 1\). Then, by definition of \(P_X\), for all \(i \geq 1\), there is \(j \in \{1, \ldots, \text{ar}(f_i)\} \notin \mu(f_i)\) such that \(u_i[j] \ni u_i\). According to Definition 5, we have that \(\sigma(u_i[j]) \ni \sigma(u_i) \ni t_i\) for some term \(s_i\) such that \(t_i \xrightarrow{\theta, \mu} \sigma(u_{i+1})\). Since root(s_i) \(\in X' \subseteq G\) and \(X' \subseteq F = \emptyset\) (Remark 4), any \(\mu\)-rewriting step is possible at the root of \(s_i\). Thus, root(s_i) = root(u_{i+1}) = f_{i+1} and \(u_{i+1} \notin \mu(f_{i+1})\). 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 \(t_i \xrightarrow{\theta, \mu} \sigma(u_{i+1}) \ni \sigma(u_{i+1})\), i.e., \(\sigma(u_i) \ni \sigma(u_{i+1})\) for all \(i \geq 1\). We get an infinite sequence \(\sigma(u_1) \ni \sigma(u_2) \ni \cdots\) which contradicts well-foundedness of \(\ni\). □

The following proposition establishes some important ‘basic’ cases of (absence of) infinite context-sensitive chains of pairs which are used later.

Proposition 9. Let \(R = (F, R)\) and \(P = (G, P)\) be TRSs and \(\mu \in M_{\rho_\cup}^\mu\).

1. If \(P = \emptyset\), then every \(\langle P, R, \mu \rangle\)-chain is empty.
Then, we have an infinite chain $(\mathcal{P}, \mathcal{R}, \mu)$-chain.

Proof.

1. Obvious, by Definition 5.

2. By contradiction. If there is an infinite $(\mathcal{P}, \mathcal{R}, \mu)$-chain, then, since there is no rule in $\mathcal{N}$, there is a substitution $\sigma$ such that

$$\sigma(u_1) \xrightarrow{\mathcal{P}, \mathcal{R}} \sigma(x_1) \xrightarrow{\mathcal{P}, \mathcal{R}} \sigma(x_2) \xrightarrow{\mathcal{P}, \mathcal{R}} \sigma(x_3) \xrightarrow{\mathcal{P}, \mathcal{R}} \cdots$$

where $x_i \in t^*$ for some terms $t_i \in T(\mathcal{F}, x)$ such that $\sigma(t_i) = C_i[t_i]$, for some $C_i \in \mathcal{I}_m$, and $\sigma$ is $\mathcal{P}$-safe in $C_i[t_i]$ such that $\text{prefix}(p_i) \subseteq \mathcal{F}$ for all $i \geq 1$. Since $s_i \not\in \text{Var}(u_i)$ and $u_i$ is not a variable, we have $u_i \not\supset s_i$; hence, $\sigma(u_i) \not\supset \sigma(s_i)$ (by stability of $\supset$) and also $\sigma(u_i) \not\supset s_i$ for all $i \geq 1$. Since $s_i$ and $\sigma(u_i)$ only differ in the root symbol, we can actually say that $s_i \not\supset s_{i+1}$ for all $i \geq 1$. Thus, we obtain an infinite sequence $s_1 \supset s_2 \supset \cdots$ that contradicts the well-foundedness of $\supset$.

3. Trivial. □

The following example shows that Proposition 9(2) does not hold for TRSs $\mathcal{P}$ with arbitrary rules.

**Example 12.** Consider $\mathcal{P} = \{ v(x) \rightarrow x, \mu(v(x)) \rightarrow v(x) \}$ together with a TRS $\mathcal{N}$ with an empty set of rules: $\mathcal{R} = \{(\mu, \emptyset)\}$. Let $\mu$ be given by $\mu(f) = \emptyset$ for all $f \in \mathcal{F}(\mathcal{R})$. Note that $\mathcal{P}_D$ consists of the pair $v(x) \rightarrow x$ because $x \in \text{Var}(v(x)) - \text{Var}_\mathcal{R}(v(x))$.

Then, we have an infinite chain

$$v(x) \xrightarrow{\mathcal{P}, \mathcal{R}} v(x) \xrightarrow{\mathcal{P}, \mathcal{R}} v(x) \xrightarrow{\mathcal{P}, \mathcal{R}} \cdots$$

Since $\mathcal{N} \not\supset \emptyset$, $v(x)$ is not an instance of any term in $\mathcal{N} \not\supset \emptyset$. 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 $\supset$ instead of $\rightarrow$ for connecting pairs. We also 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 recursive 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 $(\mathcal{D}, \mathcal{P}_D, \mathcal{R}, \mu)$-chain.

Proof. By contradiction. If $\mathcal{R}$ is not $\mu$-terminating, then there is $t \in T_{\mathcal{N}(\mathcal{R}, \mu)}$ (Lemma 2). By Theorem 1, there are rules $l_i \rightarrow t_i \in \mathcal{R}$, substitutions $\sigma_i$, and terms $t_i \in M_{\mathcal{N}(\mathcal{R}, \mu)}$ for $i \geq 1$ such that

$$t = t_0 \supset l_1 \supset \sigma_1(t_1) \xrightarrow{\mu} \sigma_2(t_1) \supset l_2 \supset \sigma_2(t_2) \xrightarrow{\mu} \cdots$$

where either (D1) $t_i = \sigma_i(s_i)$ for some $s_i$ such that $t_i \not\supset s_i$ or (D2) $t_i \not\supset \sigma_i(l_i)$ for some $l_i \not\in \text{Var}(t_i)$ and $t_i = \sigma_i(l_i)$ for some $l_i \not\in \text{Var}(t_i) - \text{Var}(l_i)$ and $t_i = \sigma_i(l_i)$ for some $l_i \not\in \text{Var}(t_i)$ and $t_i = \sigma_i(l_i)$ for some $l_i \not\in \text{Var}(t_i)$ and $t_i = \sigma_i(l_i)$ for some $l_i \not\in \text{Var}(t_i)$. Furthermore, since $t_{i-1} \supset l_i \supset \sigma_i(l_i)$ and $t_{i-1} \not\supset \sigma_i(l_i)$ and $t_i = \sigma_i(l_i)$ for all $i \geq 1$. Note that, since $t_i \in M_{\mathcal{N}(\mathcal{R}, \mu)}$, 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 root($t_i$) is not a defined symbol of $\mathcal{R}$.

First, note that $\mathcal{D}(\mathcal{R}, \mu)$ is a TRS $\mathcal{P}$ over the signature $\mathcal{V} = \mathcal{F} \cup \mathcal{P}_D$ and $\mu^D \in M_{\mathcal{F}(\mathcal{R}, \mu)}$ as required by Definition 5. Furthermore, $\mathcal{P}_D = \mathcal{D}(\mathcal{F}, \mathcal{R}, \mu)$ and $\mathcal{P}_D = \mathcal{D}(\mathcal{F}, \mathcal{R}, \mu)$. We can define an infinite minimal $(\mathcal{D}(\mathcal{R}, \mu), \mathcal{R}, \mu^D)$-chain using CSDPs $u_i \rightarrow v_i$ for $i \geq 1$, where $u_i$ and
Let $\text{DP}_R^1(\mathcal{R}, \mu) = P^1$ for $P = \text{DP}(\mathcal{R}, \mu)$. 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 $\forall \mathcal{N}\mathcal{N}(\text{DP}(\mathcal{R}, \mu)) = \emptyset$ and $\text{DP}(\mathcal{R}, \mu) = \emptyset$, then $\mathcal{R}$ is $\mu$-terminating.
3. If $\text{DP}(\mathcal{R}, \mu) = \text{DP}_R^1(\mathcal{R}, \mu)$, then $\mathcal{R}$ is $\mu$-terminating.

**Example 13.** Consider the following TRS $\mathcal{R}$ [44, Example 15]:

- $\text{add}(\text{true}, x) \rightarrow x$
- $\text{add}(\text{false}, y) \rightarrow \text{false}$
- $\text{if}(\text{true}, x, y) \rightarrow x$
- $\text{if}(\text{false}, x, y) \rightarrow y$
- $\text{first}(\text{add}(x), \text{cons}(y, z)) \rightarrow \text{cons}(y, \text{first}(x, z))$

This gives $\mathcal{R}$ is $\mu$-terminating.

Note also that $\forall \mathcal{N}\mathcal{N}(\text{DP}(\mathcal{R}, \mu)) = \emptyset$. 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 $\text{DP}(\mathcal{R}, \mu) = \text{DP}_R(\mathcal{R}, \mu)$ (both $\text{DP}(\mathcal{R}, \mu)$ and $\text{DP}_R^1(\mathcal{R}, \mu)$ are empty). We have the following infinite (DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\ast)-chain:

$$\mathcal{P}(\mathcal{L}) \hookrightarrow_{\mathcal{R}, \mu} \mathcal{P}(\text{cof}(\mathcal{L})), \mathcal{P}(\text{cof}(\mathcal{L})), \mathcal{P}(\text{cof}(\mathcal{L})), \mathcal{P}(\text{cof}(\mathcal{L})), \ldots$$

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 (DP(\mathcal{R}, \mu), \mathcal{R}, \mu^\ast)-chain.
Context-Sensitive Dependency Pairs

938

B. Alur et al. / Information and Computation 208 (2010) 922–968

Proof. By contradiction. If there is an infinite (DP(\(\mathcal{R}, \mu\)), \(\mathcal{R}, \mu^\mu\))-chain, then there are a substitution \(\sigma\) and dependency pairs \(u_i \rightarrow v_i \in \text{DP}(\mathcal{R}, \mu)\) such that

1. \(\sigma(v_i) \rightarrow_{\mu^\mu}^* \sigma(u_{i+1})\), if \(u_i \rightarrow v_i \in \text{DP}_T(\mathcal{R}, \mu)\), and
2. if \(u_i \rightarrow v_i = u_j \rightarrow x_j \in \text{DP}_T(\mathcal{R}, \mu)\), then there is \(s_i \in T(\mathcal{F}, \mathcal{X})\) such that \(\sigma(x_i) \geq_{\mu^\mu} s_i\) and \(s_i \rightarrow_{\mu^\mu} \sigma(u_{i+1})\).

for \(i \geq 1\). Now, consider the first dependency pair \(u_1 \rightarrow v_1\) in the sequence:

1. If \(u_1 \rightarrow v_1 \in \text{DP}_T(\mathcal{R}, \mu)\), then \(v_1^*\) is a \(\mu\)-replacing subterm of the right-hand-side \(r_1\) of a rule \(h_1 \rightarrow r_1\) in \(\mathcal{R}\).

   Therefore, \(r_1 = C_1(v_1^*)\) for some position \(p_1 \in \text{Pos}(r_1)\) and context \(C_1[p_1]\), and we can perform the \(\mu\)-rewriting step \(t_1 = \sigma(u_1) \rightarrow_{\mu^\mu} \sigma(r_1) = \sigma(C_1)[v_1^*]p_1 = s_1\), where \(\sigma(v_1^*) = \sigma(x_1) \rightarrow_{\mu^\mu} \sigma(u_1)\) and \(\sigma(u_1)\) initiates an infinite \((\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\mu)\)-chain. Note that \(p_1 \in \text{Pos}(r_1)\).
2. If \(u_1 \rightarrow x \in \text{DP}_T(\mathcal{R}, \mu)\), then there is a rule \(h_1 \rightarrow r_1\) in \(\mathcal{R}\) such that \(u_1 = \ell_1\) and \(x \in \text{Var}^\mu_C(r_1) \cup \text{Var}^\mu_C(l_i)\), i.e., \(r_1 = C_1[\ell_1]\) for some \(q_1 \in \text{Pos}(r_1)\). Furthermore, since \(\sigma(x) = C_1[\ell_1]\) for some term \(C_1[\ell_1]\), and \(p_1 \in \text{Pos}(C_1[\ell_1])\) such that \(p_1 \rightarrow_{\mu^\mu} \sigma(u_1)\), we can perform the \(\mu\)-rewriting step \(t_1 = \sigma(l_i) \rightarrow_{\mu^\mu} \sigma(r_1) = \sigma(C_1)[v_1^*]p_1 = s_1\)

   where \(p_1 \rightarrow_{\mu^\mu} \sigma(u_1)\) (hence \(s \rightarrow_{\mu^\mu} \sigma(u_1)\)) and \(\sigma(u_1)\) initiates an infinite \((\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\mu)\)-chain. Note that \(p_1 \in \text{Pos}(\ell_1)\) (use Proposition 1).

Since \(\mu^\mu(f^2) = \mu(f)\), and \(p_1 \in \text{Pos}(\ell_1)\), we have that \(s_1 \rightarrow_{\mu^\mu} t_2 \rightarrow_{\mu^\mu} \cdots \rightarrow_{\mu^\mu} \cdots\) which contradicts the \(\mu\)-termination of \(\mathcal{R}\). □

Proposition 9(3) suggests a simple checking of non-\(\mu\)-termination.

Corollary 4 (Non-\(\mu\)-termination criterion). Let \(\mathcal{R} = (\mathcal{F}, \mathcal{R}, \mu)\) be a TRS and \(\mu \in M_2\). If there is \(u \rightarrow v \in \text{DP}_T(\mathcal{R}, \mu)\) 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 \(\mu \in M_2\). Then, \(\mathcal{R}\) is \(\mu\)-terminating if and only if there is no infinite minimal \((\mathcal{DP}(\mathcal{R}, \mu), \mathcal{R}, \mu^\mu)\)-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 \(\text{DP}(\mathcal{R})\) is obtained. Then, these dependency pairs are organized in a dependency graph \(\text{DG}(\mathcal{R})\) whose nodes are the pairs in \(\text{DP}(\mathcal{R})\) 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., \(u_i \rightarrow v_i \in \text{DP}(\mathcal{R})\) 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 mechanism 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 \(\mathcal{P} = \text{DP}(\mathcal{R})\): it 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.

\footnote{The original definition in [35] includes an extra parameter \(v\), which specifies two kinds of problems: \(v = 1\) for termination problems, and \(v = 0\) for instance termination problems.}
7.1. CS problems, CS processors, and the CSDP-framework

In our definition of DP problem for CSR, we prefer to avoid ‘DP’ because, as discussed above, dependency pairs (as such) are relevant in the theoretical framework only for investigating a particular problem (termination of TRSs), whereas some transformations can yield sets of pairs which are no longer dependency pairs of the underlying TRS.

Definition 6 (CS problem). A CS problem τ is a tuple (P, R, μ), where R and P are TRSs and μ ∈ M_{R∪P}. The CS problem τ is finite if there is no infinite minimal (P, R, μ)-chain. The CS problem τ is infinite if R is non-μ-terminating or there is an infinite minimal (P, R, μ)-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 R is nonterminating as part of the definition of infinite problem is essential for dealing with some specific transformations of CS problems (see Theorems 8 and 16).

Definition 7 (CS processor). A CS processor Proc is a mapping from CS problems into sets of CS problems. Alternatively, it can also return “no”. A CS processor Proc is

- sound if for all CS problems τ, we have that (1) τ is finite whenever Proc(τ) ̸= no and (2) ∀τ′ ∈ Proc(τ), τ′ is finite.
- complete if for all CS problems τ, we have that (1) τ is finite whenever Proc(τ) = no or (2) ∃τ′ ∈ Proc(τ) such that τ′ is infinite.

A (sound) processor transforms DP problems into (hopefully) simpler ones, in such a way that the existence of an infinite chain in the original DP problem implies the existence of an infinite chain in the transformed one. Here, ‘simpler’ usually means that fewer pairs are involved. Soundness is essential for proving termination. Completeness is necessary for proving nontermination.

Processors are used in a divide and conquer scheme to incrementally simplify the original CS problem as much as possible, possibly decomposing it into smaller pieces which are then independently treated in the very same way. The trivial case comes when the set of pairs 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 R be a TRS and μ ∈ M_{R}. We construct a tree whose nodes are labeled with CS problems or “yes” or “no”, and whose root is labeled with (DP(R, μ), R, μ^2). For every inner node labeled with τ, there is a sound processor Proc satisfying one of the following conditions:

1. Proc(τ) = no and the node has just one child that is labeled with “no”.
2. Proc(τ) = δ and the node has just one child that is labeled with “yes”.
3. Proc(τ) ̸= no, Proc(τ) ̸= δ, and the children of the node are labeled with the CS problems in Proc(τ).

If all leaves of the tree are labeled with “yes”, then R is μ-terminating. Otherwise, if there is a leaf labeled with “no” and if all processors used on the path from the root to this leaf are complete, then R is not μ-terminating.

Propositions 8 and 9 are the basis for the following sound and complete processors, which provide some base cases for our proofs of termination of CSR.

Theorem 5 (Basic CS processors). Let R = (F, R) and P = (g, P) be TRSs and μ ∈ M_{R∪P}. Then, the processors Proc_{CS} and Proc_{CSD} given by

\[
\text{Proc}_{\text{CS}}(P, R, μ) = \begin{cases} 
\emptyset & \text{if } P = \emptyset \lor P = P^1 \lor (R = \emptyset \land P = P^2); \\
\{(P, R, μ)\} & \text{otherwise} 
\end{cases}
\]

\[
\text{Proc}_{\text{CSD}}(P, R, μ) = \begin{cases} 
n & \text{if } \nu = θ(u) \\
\{(P, R, μ)\} & \text{otherwise} 
\end{cases}
\]

are sound and complete.

\[^{5}\text{In the following, we often write Proc(P, R, μ) instead of Proc((P, R, μ)) to avoid duplicated parentheses.}\]
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 \( \text{DG}(R) \) is associated to the TRS \( R \). The nodes of \( \text{DG}(R) \) are the dependency pairs in \( \text{DP}(R) \); there is an arc from a dependency pair \( u \rightarrow v \) to a dependency pair \( u' \rightarrow v' \) such that \( \var(u) \cap \var(u') = \emptyset \) if \( \theta(v) \rightarrow^{*}_{\tau} \theta(u') \) for some substitution \( \theta \). In [35], a more general notion of graph of pairs \( \text{DG}(P, R) \) associated to a set of pairs \( P \) and a TRS \( R \) is considered. Pairs in \( P \) are now used as the nodes of the graph, but they are connected by \( R \)-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 \( R \) and \( P \) and a replacement map \( \mu \in M_{EUP} \), 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 \( R \) and \( P \) be TRSs and \( \mu \in M_{EUP} \). The context-sensitive (CS)-graph \( G(P, R, \mu) \) has \( P \) as the set of nodes. Given \( u \rightarrow v, u' \rightarrow v' \in P \), there is an arc from \( u \rightarrow v \) to \( u' \rightarrow v' \) if \( u \rightarrow v, u \rightarrow v' \) is a minimal \( (P, R, \mu) \)-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) [38]. 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 \( (P, R, \mu) \)-chains.

**Theorem 6** (SCC processor). Let \( R \) and \( P \) be TRSs and \( \mu \in M_{EUP} \). Then, the processor \( \text{Proc}_{SCC} \) given by

\[
\text{Proc}_{SCC}(P, R, \mu) = \{ (Q, R, \mu) \mid Q \text{ are the pairs of an SCC in } G(P, R, \mu) \}
\]

is sound and complete.

**Proof.** We prove soundness by contradiction. Assume that \( \text{Proc}_{SCC} \) is not sound. Then, there is a CS problem \( \tau = (P, R, \mu) \) such that, for all \( \tau' \in \text{Proc}_{SCC}(\tau) \), \( \tau' \) is finite but \( \tau \) is not finite. Thus, there is an infinite minimal \( (P, R, \mu) \)-chain \( A \). Since \( P \) contains a finite number of pairs, there is \( P' \subseteq P \) and a tail \( B \) of \( A \), which is an infinite minimal \( (P', R, \mu) \)-chain where all pairs in \( P' \) are infinitely often used. According to Definition 8, this means that \( P' \) is a cycle in \( G(P', R, \mu) \). Hence \( P' \) belongs to some SCC with nodes in \( Q \), i.e., \( P' \subseteq Q \). Thus, \( B \) is an infinite minimal \( (Q, R, \mu) \)-chain, i.e., \( \tau' = (Q, R, \mu) \) is not finite. Since \( \tau' \in \text{Proc}_{SCC}(\tau) \), we obtain a contradiction.

With regard to completeness, since \( Q \subseteq P \) for some SCC in \( G(P, R, \mu) \) with nodes in \( Q \), every infinite minimal \( (Q, R, \mu) \)-chain is an infinite minimal \( (P, R, \mu) \)-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 \( G(P, R, \mu) \), 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 \( R \) be a TRS and \( \mu \in M_{E} \). The Context-Sensitive Dependency Graph (CSDG) for \( R \) and \( \mu \) is \( G(DP(R, \mu)), R, \mu \).

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 \( u \rightarrow v \in P_{\tau} \)) or \( \sigma(t') \) (for \( t \in \text{NT}(\tau) \)) 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 \( R \) and a replacement map \( \mu \), let \( \text{TCP}_{\mu} \) be as follows:
where $y$ is intended to be a new, fresh variable that has not yet been used and given a term $s, [s]^f_i = \text{TCAP}^\mu(s)$ if $i \in \mu(f)$ and $[s]^f_i = s$ if $i \notin \mu(f)$. We assume that $t$ shares no variable with $f([t_1]^f_i, \ldots, [t_n]^f_i)$ when the unification is attempted. Function $\text{TCAP}^\mu(s)$ is intended to provide a suitable approximation of the aforementioned $(R, \mu)$-reachability problems by means of unification. The following result formalizes the correctness of this approach.

**Proposition 10.** Let $R = (F, R)$ be a TRS and $\mu \in M_F$. Let $t, u \in T(F, R)$ be such that $\text{Var}(t) \cap \text{Var}(u) = \emptyset$. If $\theta(t) \rightarrow \ast \theta(u)$ for some substitution $\theta$, then $\text{TCAP}^\mu(t)$ and $u$ unify.

**Proof.** In the following, we let $s = \text{TCAP}^\mu(t)$. 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 $\theta(u) = \theta(\sigma(u))$, i.e., $s$ and $u$ unify.
2. If $m > 0$, then we have $\theta(t) \rightarrow t' \rightarrow \ast \theta(u)$. Let $p \in \text{Par}^\mu(\theta(t))$ be the position where the $\mu$-rewrite step $\theta(t) \rightarrow t'$ is performed. By definition of $\text{TCAP}^\mu$, $s = [z]^f_q$ for some fresh variable $z$ and position $q$ such that $q \leq p$. We can write $\theta(t) = \theta(u)$. Furthermore, since $z$ is a fresh variable, we can write $t' = \theta(s)$ if we assume that $\theta(z) = t^\ast$. Thus, $\theta(s) \rightarrow \ast \theta(u)$ in $m - 1$ steps. By the induction hypothesis, $\text{TCAP}^\mu(t)$ and $u$ unify. Since $\text{TCAP}^\mu(t) = \text{TCAP}^\mu(\text{TCAP}^\mu(t))$ and $\text{TCAP}^\mu(t)$ is just a renaming of $\text{TCAP}^\mu(t)$, the conclusion follows.

According to Proposition 10, given terms $t, u \in T(F, R)$ 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 $\text{TCAP}^\mu(t)$ and $u$. Thus, taking into account Definitions 5 and 8, we have the following.

**Definition 10.** (Estimated context-sensitive graph of pairs). Let $R$ and $P$ be TRSs and $\mu \in M_G$ be a context. The estimated CS-graph associated to $R$ and $P$ (denoted $\text{EG}(R, P)$) has $P$ as the set of nodes and the arcs that connect them as follows:

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

As a consequence of Proposition 10, we have the following.

**Corollary 6.** (Approximation of the context-sensitive graph). Let $R$ and $P$ be TRSs and $\mu \in M_G$. The estimated CS-graph $\text{EG}(P, R, \mu)$ contains the CS-graph $G(P, R, \mu)$.

Therefore, we have the following estimated CSDG: $\text{EDG}(R, \mu) = \text{EG}(\text{OP}(R, \mu), R, \mu)$.

**Remark 10.** Proposition 10 also provides estimations for $NNP$: if $t \in NNP$, then $\text{TCAP}^\mu(t)$ and $u$ unify for some $u \rightarrow v \in P$. In the following, we compute $NNP$ in this way.

![Fig. 5](image-url) Context-sensitive dependency graph for the CS-TRS in Example 1.
Example 15. Consider again the CS-TRS \((R, \mu)\) in Example 1. Note that
\[ NHT_{DP}(\mu, \mu) = \{ \text{oDDia}, \text{incr(oDDia)}, \text{incr(x)}, \text{zip(x, y)}, \text{rep(z(x))} \} \]

The (estimated) CSG in Fig. 5 has four cycles, each of which contains a single pair. We transform the CS problem
\((DP(R, \mu), R, \mu)\) into a set
\[ \text{Proc}_{CS}(DP(R, \mu), R, \mu) = \left\{ \left(\{1\}, R, \mu^1\right), \left(\{17\}, R, \mu^1\right), \left(\{21\}, R, \mu^1\right), \left(\{23\}, R, \mu^1\right) \right\} \]

which contains four new (but very simple) CS problems.

Remark 11 (CSG vs. DG). Consider again \(R\) and \(\mu\) as in Example 1. Pairs (9) and (10) belong to both DG(\(R\)) (see Fig. 3) and DG(\(R, \mu\)) (see Fig. 5). However, they are not equally connected in DG(\(R\)) and DG(\(R, \mu\)). The reason is that the collapsing pair (25), that is not a node of DG(\(R\)), 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 \(R = (F, R)\) and \(P = (G, P)\) be TRSs and \(\mu \in M_{R,DG}\). Let \(P' = (G', P')\) where \(P' = (P - P_{\epsilon}) \cup Q\) for \(Q = \{ u \rightarrow t^2 \mid u \in R, t \in NHT_{R} \}\). If \(G' = G\) if \(Q = \emptyset\), and \(G' = F \cup G\) if \(Q \neq \emptyset\). Then, the processor \(\text{Proc}_{CS}(P', R, \mu)\) is sound.

**Proof.** First, note that \(P'\) is a TRS: the new rules in \(Q\) are of the form \(u \rightarrow t^2\) for \(t \in NHT_{R}\). Since \(NHT_{R} \subseteq T(F)\), we trivially have \(\forall u \in \text{Var}(F) \subseteq \text{Var}(u)\), i.e., \(u \rightarrow t^2\) is a rewrite rule. Furthermore, whenever \(Q \neq \emptyset\), \(G'\) is the union of \(F\) and \(G\) to reflect the use of symbols in \(F\) coming from terms \(t^2\) for \(t \in NHT_{R}(R, \mu)\). Since we assume that \(\mathcal{D}^\Phi \subseteq G\) (Remark 4), \(\omegaPF \subseteq \mathcal{D}^\Phi \subseteq G\).

We prove that the existence of an infinite minimal \((P', R, \mu)\)-chain implies the existence of an infinite minimal \((P', R, \mu)\)-chain. Consider an infinite minimal \((P', R, \mu)\)-chain:

\[ \sigma(u_1) \rightarrow_{P', \mu} 0 \circ \tilde{P}_{\mu} \circ \tilde{t}_1 \rightarrow_{P', \mu} \sigma(u_2) \rightarrow_{P', \mu} 0 \circ \tilde{P}_{\mu} \circ \tilde{t}_2 \rightarrow_{P', \mu} \sigma(u_3) \rightarrow_{P', \mu} 0 \circ \tilde{P}_{\mu} \circ \cdots \]

for some substitution \(\sigma\), where, according to Definition 5, for all \(i \geq 1\), \(t_i\) is \(\mu\)-terminating and, (1) if \(u_i \rightarrow v_i \in \mathcal{P}_F\), then \(\sigma(u_i) \rightarrow_{\mu} v_i \in \mathcal{P}_F\), then \(t_i = \sigma(u_i)\) for some \(u_i\), such that \(\sigma(u_i) \circ \mathcal{P}_F\), and \(v_i = \hat{\theta}(t_i)\) for some \(\hat{\theta} \in NHT\) and substitution \(\hat{\theta}\). Actually, since \(\hat{\theta}(t_i) = \hat{\theta}(\hat{\theta}(t_i)) = \tilde{t}_i\), we can further say that \(\tilde{t}_i \in NHT\).

In case (2), since \(NHT_{R} \subseteq T(F)\), we have \(t_i = \sigma(u_i)\) for all \(i \geq 1\), i.e., \(t_i \in NHT_{R}\). Thus, we can use \(u_i \rightarrow v_i \in \mathcal{Q}\) instead of \(u_i \rightarrow v_i \in \mathcal{P}_F\), because we still have \(t_i \rightarrow_{P', \mu} \sigma(u_{i+1})\). In this way, by replacing each \(u_i \rightarrow v_i \in \mathcal{Q}\) by the corresponding \(u_i \rightarrow \tilde{t}_i \in Q\), each step \(\sigma(u_i) \rightarrow_{P', \mu} 0 \circ \tilde{P}_{\mu} \circ \tilde{t}_i \) becomes a step \(\sigma(u_i) \rightarrow_{P', \mu} 0 \circ \tilde{P}_{\mu} \circ \tilde{t}_i\), whereas steps \(\sigma(u_i) \rightarrow_{P', \mu} \sigma(v_i) = t_i\) for \(u_i \rightarrow v_i \in \mathcal{Q}\) remain unchanged. Thus, we obtain an infinite minimal \((P', R, \mu)\)-chain, as desired. \(\square\)

Note that no pair in \(P'\) in Theorem 7 is collapsing. Unfortunately, \(\text{Proc}_{CS}(P', R, \mu)\) is not complete.

Example 16. Consider the following TRS:

\[ b \rightarrow \text{f(c(b))} \]
\[ \text{f}(x) \rightarrow x \]

together with the replacement map \(\mu\) given by \(\mu(\text{f}) = \mu(v) = \emptyset\). \(\text{DP}(R, \mu)\) is:

\[ b \rightarrow \text{f(c(b))} \]
\[ \text{f}(x) \rightarrow x \]
and \( \mathcal{N}(\mathcal{D}(P, R, \mu)) = \{ 0 \} \). There is no infinite \((P, R, \mu^2)\)-chain for \( P = \mathcal{D}(R, \mu) \), i.e., \( \mathcal{D}(R, \mu) \) is finite and \( R \mu^2 \)-terminating. However, with \( P' \) as in Theorem 7:

\[
B \rightarrow F(c(b)) \\
F(x) \rightarrow B
\]

we have an infinite minimal \((P', R, \mu^2)\)-chain, i.e., \((P', R, \mu^2)\) is not finite.

The following processor provides a sound and complete transformation of collapsing pairs into noncollapsing pairs.

**Theorem 8** (Transforming collapsing pairs). Let \( R = (F, R) \) and \( P = (G, P) \) be TRSs and \( \mu \in M_{R, G} \), Let \( u \rightarrow x \in P_R \) and \( P_u = \{ u \rightarrow U(x) \} \)

\[
\begin{align*}
\cup \{ U(F(x_1, \ldots, x_n)) \rightarrow U(x_i) \mid F \in F, i \in \mu[F] \} \\
\cup \{ U(t) \rightarrow t' \mid t \in \mathcal{N}(T_F) \}
\end{align*}
\]

where \( U \) is a fresh symbol. Let \( P' = (G \cup \{ U \}, P') \) where \( P' = (P - \{ u \rightarrow x \}) \cup P_u \) and \( \mu' \) which extends \( \mu \) by \( \mu' \) (\( U(x) \)). The processor \( \text{ProcrCal} \) given by

\[
\text{ProcrCal}(P, R, \mu) = \{(P', R, \mu')\}
\]

is sound and complete.

**Proof.** With regard to soundness, we proceed by contradiction. If \( \text{ProcrCal} \) is not sound, then there is an infinite minimal \((P', R, \mu')\)-chain \( \sigma \) such that \( P' \) has no \((P', R, \mu')\)-chain. Since \( P' \) is finite, we can assume that there is \( \varnothing \subseteq P \) such that \( A \) has a tail \( B \)

\[
\sigma(x_1) \xrightarrow{\lambda_{\varnothing, \mu'}} \cdots \xrightarrow{\lambda_{\varnothing, \mu'}} t_1 \xrightarrow{\varnothing, \mu'} t_2 \xrightarrow{\varnothing, \mu'} \cdots
\]

for some substitution \( \sigma \) and pairs \( u_i \rightarrow x_i \). Then for all \( i \geq 1 \),

1. If \( v_i \not\in X \), then \( t_i = \sigma(v_i) \).
2. If \( v_i \in X \), then \( t_i = \varnothing u_i \) and \( \sigma(v_i) = G[s_i] \) for some context \( G[s_i] \), such that \( p_i \in \mathcal{D}(G[s_i]) \).

For ‘steps’ \( \sigma(x_i) \xrightarrow{\lambda_{\varnothing, \mu'}} \cdots \xrightarrow{\lambda_{\varnothing, \mu'}} t_1 \xrightarrow{\varnothing, \mu'} t_2 \xrightarrow{\varnothing, \mu'} \cdots \)

where all terms of the form \( U(t) \) in the sequence above are \((R, \mu')\)-terminating: since \( \mu'(U) = \varnothing \) and \( U \) does not belong to \( F, U \) is \((R, \mu')\)-normal form. Furthermore, by minimality of \( R, t_i \) is \((R, \mu')\)-terminating and, since \( \mu'(f) = \mathcal{D}(F, C_1) \) for all \( f \in \mathcal{N}(R, \mu'), \mu'(R, \mu') \)-chain, leading to a contradiction.

For completeness, we consider two cases: if \( R \) is not \( \mu' \)-terminating, then all termination problems are infinite (both before and after the application of \( \text{ProcrCal} \)) and there is no problem. Therefore, assume that \( R \) is \( \mu' \)-terminating and that \((P', R, \mu')\) is finite but there is an infinite \((P', R, \mu')\)-chain. Again, we can assume that there is \( \varnothing \subseteq P' \) such that \( A \) has a tail \( B \)

\[
\sigma(x_1) \xrightarrow{\lambda_{\varnothing, \mu'}} \cdots \xrightarrow{\lambda_{\varnothing, \mu'}} t_1 \xrightarrow{\varnothing, \mu'} \cdots \xrightarrow{\lambda_{\varnothing, \mu'}} t_2 \xrightarrow{\varnothing, \mu'} \cdots
\]

for some substitution \( \sigma \) and pairs \( u_i \rightarrow v_i \) where \( t_i = \sigma(v_i) \) is \((R, \mu')\)-terminating for \( i \geq 1 \). Without loss of generality, we can assume that \( \sigma(x) \in \mathcal{N}(R, \mu') \) for all \( x \in X \), i.e., \( \sigma \) does not introduce any symbol \( U \). It is not difficult to see that, for each \((P', R, \mu')\)-chain which is based on a substitution \( \sigma' \) whose bindings \( \sigma'(x) \) contain symbols \( U \), there is a \((P', R, \mu')\)-chain which uses the same pairs in \( P' \) and rules in \( R \) 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 \( \mu(U) = \varnothing \), there is \( n \geq 0 \) such that
Context-Sensitive Dependency Pairs

944 B. Alarcón et al. / Information and Computation 208 (2010) 922–968

\[ \sigma(u_i) \rightarrow^{P, \mu'} \sigma(U(x_i)) = U(\sigma(x_i)) = \sigma(u_{i+1}) \]
\[ \rightarrow^{P, \mu'} \sigma(u_{i+1}) = \sigma(u_{i+2}) \]
\[ \vdots \]
\[ \rightarrow^{P, \mu'} \sigma(u_{i+n}) = \sigma(U(s_{i+n+1})) \]
\[ \rightarrow^{P, \mu'} \sigma(s_{i+n+1}) = \sigma(t_i) \]
\[ \rightarrow^{P, \mu'} \sigma(u_{i+n+2}) \]

where, for all \( i, j \in 1 \leq j \leq i+n \), \( u_i = U(f_i(x_1, \ldots, x_n)) \), \( v_j = U(x_i) \), \( t_i \in \mu(f) \), and \( s_{i+n+1} \in N^N \mid T \) (by definition of \( \text{ProcGal} \)). Therefore, from the \( n \) rewriting steps that remove the \( f_j \in \mathcal{F} \) for \( 1 \leq j \leq n \), we know that \( \sigma(x) = C_l[t_{i+n+1}] \) with \( p_i \in \text{ProcGal}(C_l \mid t_{i+1}) \) and \( \text{specixo}_l(p_i) \subseteq \mathcal{F} \). Thus, according to Definition 5, we have: \( \sigma(u_i) \rightarrow^{P, \mu} \circ \circ^{\mu} t_i \) and \( t_i \rightarrow^{\mu, \mu} \sigma(u_{i+n+2}) \). Furthermore, \( t_i \) is \( \mu \)-terminating (because \( \mathcal{R} \) is \( \mu \)-terminating). On the other hand, if \( u_n \rightarrow v_i \in \mathcal{P} \) \( - \{ x \rightarrow \} \), then we have \( \sigma(u_i) \rightarrow^{P, \mu} \circ \circ^{\mu} t_i \) satisfying the conditions in Definition 5. We obtain an infinite minimal \( (P, \mathcal{R}, \mu) \)-chain, leading again to a contradiction. \( \square \)

Example 17. The use of \( \text{ProcGal} \) with \( (D\mathcal{P}(\mathcal{R}, \mu), \mathcal{R}, \mu^2) \) in Example 16 yields \( (P', \mathcal{R}, \mu') \) where \( P' \) consists of the following pairs:
\[
\begin{align*}
\mathcal{P} & \rightarrow P(c(b)) \\
& \rightarrow P(x) \rightarrow \mu(x) \\
& \rightarrow \mu(b) \rightarrow b
\end{align*}
\]
It is not difficult to see now that there is no infinite minimal \( (P', \mathcal{R}, \mu') \)-chain.

10. Use of \( \mu \)-reduction pairs

A reduction pair \( (\geq, \sqsupseteq) \) consists of a stable and monotonic quasi-ordering \( \geq \), and a stable and well-founded ordering \( \sqsupseteq \) satisfying either \( \geq \circ \sqsubseteq \sqsupseteq \circ \sqsubseteq \sqsubseteq \sqsubseteq \sqsubseteq \) [42]. The absence of infinite chains of pairs can be ensured by finding a reduction pair \( (\geq, \sqsupseteq) \) that is compatible with the rules and the pairs: \( l \geq r \) for all rewrite rules \( l \rightarrow r \) and \( u \sqsubseteq v \) for all dependency pairs \( u \rightarrow v \) [10]. In the dependency pair framework, they are used to obtain smaller sets of pairs \( P' \subseteq P \) by removing the strict pairs, i.e., those pairs \( u \rightarrow v \in P \) such that \( u \sqtriangleright v \).

Stability is required for both \( \geq \) and \( \sqsupseteq \) because, although we only check the left- and right-hand sides of the rewrite rules \( l \rightarrow r \) (with \( \geq \)) and pairs \( u \rightarrow v \) (with \( \sqsubseteq \)), the chains of pairs involve instances \( \sigma(l), \sigma(t), \sigma(u), \) and \( \sigma(v) \) of rules and pairs, and we aim to conclude \( \sigma(l) \geq \sigma(r) \) and also \( \sigma(u) \sqsubseteq \sigma(v) \). Monotonicity is required for \( \geq \) 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 \( \sqsupseteq \) (but, for this reason, we cannot compare the rules in \( \mathcal{R} \) using \( \sqsupseteq \)). Endrullis et al. noticed that transitivity is not necessary for the strict component \( \geq \) 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 \( \mu \in \mathcal{M}_F \). A \( \mu \)-reduction pair \( (\geq, \sqsupseteq) \) consists of a stable and \( \mu \)-monotonic quasi-ordering \( \geq \) and a well-founded stable relation \( \sqsupseteq \) on terms in \( \mathcal{T}(\mathcal{F}, \mathcal{X}) \) that are compatible, i.e., \( \geq \circ \sqsubseteq \sqsubseteq \sqsubseteq \sqsubseteq \). We say that \( (\geq, \sqsupseteq) \) is \( \mu \)-monotonic if \( \sqsupseteq \) 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 \( \mu \in \mathcal{M}_F \). Let \( (\geq, \sqsupseteq) \) be a \( \mu \)-monotonic \( \mu \)-reduction pair such that \( \geq \) for all \( l \rightarrow r \in \mathcal{R} \). Let \( \mathcal{R}_{\leq} = \{ l \rightarrow r \in \mathcal{R} | l \geq r \} \) and \( S = \mathcal{R} - \mathcal{R}_{\leq} \). Then, \( \mathcal{R} \) is \( \mu \)-terminating if and only if \( S \) is \( \mu \)-terminating.

Proof. Since \( \mathcal{S} \subseteq \mathcal{R} \), the only if part is obvious. For the if part, we proceed by contradiction. If \( \mathcal{R} \) is not \( \mu \)-terminating, then there is an infinite \( \mu \)-rewrite sequence \( \Lambda \):}
\[
\begin{align*}
l_1 \rightarrow^{\mu, \mu} l_2 \rightarrow^{\mu, \mu} \cdots l_n \rightarrow^{\mu, \mu} \cdots
\end{align*}
\]
where an infinite number of rules in $\mathcal{R}_1$ have been used; otherwise, there would be an infinite tail $t_{m} \leftarrow s, \mu_{t_{m+1}} \leftarrow s, \mu_{t_{m+2}} \leftarrow s, \mu_{t_{m+3}} \leftarrow s, \mu_{t_{m+4}} \leftarrow s, \cdots$ for some $m \geq 3$ where only rules in $\mathcal{S}$ are applied, contradicting the $\mu$-termination of $\mathcal{S}$. Let $f = [t_1, t_2, \ldots]$ be the infinite set of indices indicating $\mu$-rewriting steps $t_i \leftarrow s$ in $\mathcal{A}$, for all $j \in J$, where rules in $\mathcal{R}_1$ have been used to perform the $\mu$-rewriting step. Since $f \not\Rightarrow r$ for all $t \rightarrow r$ in $\mathcal{R}_2$, by stability and $\mu$-monotonicity of $\mathcal{S}$, we have that $t_{i_1} \not\Rightarrow t_{i_2}$. Since $f \not\Rightarrow r$ for all $t \rightarrow r$ in $\mathcal{S}$, by stability and $\mu$-monotonicity of $\mathcal{S}$, we have that $t_{i_{j+1}} \not\Rightarrow t_{i_j}$. By compatibility between $\mathcal{S}$ and $\mathcal{T}$, we have $t_{i_j} \not\Rightarrow t_{i_{j+1}}$, for all $i \geq 1$. We obtain an infinite sequence $t_{i_1} \not\Rightarrow t_{i_2} \not\Rightarrow \cdots$ which contradicts well-foundedness of $\mathcal{S}$.

10.1. Argument filterings for CSR

An argument filtering $\pi$ for a signature $\mathcal{S}$ is a mapping that assigns to every $k$-ary function symbol $f \in \mathcal{S}$ an argument position $i \in \{1, \ldots, k\}$ or a (possibly empty) list $[i_1, \ldots, i_m]$ of argument positions with $1 \leq i_1 < \cdots < i_m \leq k$ [42]. The trivial argument filtering $\pi$ is given by $\pi(f) = [1, \ldots, k]$ for each $k$-ary symbol $f \in \mathcal{S}$. 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 $\mathcal{S}$ or any replacement map $\mu \in \mathcal{M}_\mathcal{S}$. In the following, we assume that:

1. The signature $\mathcal{S}_\pi$ consists of all function symbols $f$ such that $\pi(f)$ is some list $[i_1, \ldots, i_m]$, where, in $\mathcal{S}_\pi$, the arity of $f$ is $m$.
2. As usual, we give the same name to the version of $f \in \mathcal{S}$ that belongs to $\mathcal{S}_\pi$.
3. The replacement map $\mu_\pi \in \mathcal{M}_\mathcal{S}_\pi$ is given as follows: for all $f \in \mathcal{S}$ such that $f \in \mathcal{S}_\pi$ and $\pi(f) = [i_1, \ldots, i_m]$:

   $\mu_\pi(f) = \{i \in \{1, \ldots, m\} \mid i \in \pi(f)\}$

An argument filtering $\pi$ induces a mapping from $\mathcal{T}(\mathcal{S}, \mathcal{X})$ to $\mathcal{T}(\mathcal{S}_\pi, \mathcal{X})$, also denoted by $\pi$:

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

Note that, for the trivial argument filtering $\pi$, we have that $\mathcal{S}_\pi = \mathcal{S}$ and $\mu_\pi = \mu$ for all $\mu \in \mathcal{M}_\mathcal{S}$. Furthermore, $\pi_{\mathcal{S}_t}(t) = t$ for all $t \in \mathcal{T}(\mathcal{S}, \mathcal{X})$. 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 $\mathcal{S}$ be a signature, $\pi$ be an argument filtering for $\mathcal{S}$, and $\sigma$ be a substitution. If $t \in \mathcal{T}(\mathcal{S}_\pi, \mathcal{X})$, 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 $\pi(t) = x$ and $\pi(\sigma(t)) = \sigma_\pi(x) = \sigma_\pi(\pi(t))$. If $t$ is a constant symbol, then $\pi(\sigma(t)) = t$ and $\pi(t) = t = \sigma_\pi(t) = \pi(\sigma(t))$.
2. If $t = f(t_{i_1}, \ldots, t_{i_k})$, then we consider the two possible cases according to $\pi(f)$:
   a. If $\pi(f) = i$ for some $i \in \{1, \ldots, k\}$, then $\pi(t) = \pi(t_i)$. By the induction hypothesis, $\pi(\sigma(t_i)) = \sigma_\pi(\pi(t_i))$. Therefore, $\pi(\sigma(t)) = \pi(f(\sigma(t_{i_1}), \ldots, \sigma(t_{i_k}))) = \pi(\sigma(t_i)) = \sigma_\pi(\pi(\sigma(t_i))) = \sigma_\pi(\pi(\pi(t_i))) = \sigma_\pi(\pi(\pi(t)))$.
   b. If $\pi(f) = [i_1, \ldots, i_m]$, then $\pi(t) = f(\pi(t_{i_1}), \ldots, \pi(t_{i_m}))$. By the induction hypothesis, $\pi(\sigma(t_{i_j})) = \sigma_\pi(\pi(\sigma(t_{i_j})))$, for all $j \in \{1, \ldots, m\}$. Thus, $\pi(\sigma(t)) = \pi(f(\sigma(t_{i_1}), \ldots, \sigma(t_{i_m}))) = \sigma_\pi(\pi(\pi(t))$, $\cdots, \sigma_\pi(\pi(\pi(t_{i_m})))) = \sigma_\pi(\pi(\pi(t)))$. $\square$

**Proposition 12.** Let $\mathcal{R} = (\mathcal{S}, \mathcal{R})$ be a TRS, $\mu \in \mathcal{M}_\mathcal{S}$, $\pi$ be an argument filtering for $\mathcal{S}$, and $s, t \in \mathcal{T}(\mathcal{S}, \mathcal{X})$. Let $\mu_{\mathcal{S}_t}$ be a $\mu_{\mathcal{S}_t}$-monotonic quasi-ordering such that $\pi(l) \supseteq \pi(r)$ for all $l \rightarrow r \in \mathcal{R}$. If $s \rightarrow^* t$, then $\pi(s) \supseteq \pi(t)$.

**Proof.** By induction on the length $n$ of the $\mu$-rewriting sequence.

1. If $n = 0$, then $s = t$ and, trivially, $\pi(s) = \pi(t)$. Since $\supseteq$ is reflexive, we have $\pi(s) \supseteq \pi(t)$.
2. If $n > 0$, we can write $s \rightarrow s' \rightarrow^* t$, where the length of the sequence from $s'$ to $t$ is $n - 1$. Let $p \in \text{Pos}^\circ(s)$ be the $\mu$-replacing position where the $\mu$-rewriting step $s \rightarrow s'$ is performed. We prove that $s \rightarrow s'$ implies $\pi(s) \supseteq \pi(s')$ by induction on the structure of $p$. 
Context-Sensitive Dependency Pairs

(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 \( \pi(l) \geq \pi(r) \) by stability of \( \preceq \), we conclude \( \pi(s) \geq \pi(s') \). Therefore, \( \pi(s) \geq \pi(s') \). We connect the marked version \( \pi(s) \geq \pi(s') \).

(b) If \( p = \iota \cdot q \), then we can write \( s = f(s_1, \ldots, s_k) \) and \( s' = f(s'_1, \ldots, s'_k) \) for some nonconstant symbol \( f \) (i.e., \( \alpha(f) > 0 \)) and we know that \( i \in \mu(f) \), \( s_i \rightarrow s'_i \) at position \( q \) and \( s_j = s'_j \) for all \( j \neq i \). By the induction hypothesis, \( \pi(s) \geq \pi(s') \). We consider the two possible cases according to \( \pi(f) \):

i. If \( \pi(f) = j \) for some \( j \in \{1, \ldots, k\} \), then \( \pi(s) = \pi(s') \). If \( i \neq j \), then \( s_i = s'_i \). By reflexivity of \( \preceq \), we have \( \pi(s) \geq \pi(s') \). If \( i = j \), then we know from above that \( \pi(s) \geq \pi(s') \). Therefore, \( \pi(s) = \pi(s') \). We consider \( i \) for some \( j \in \{1, \ldots, m\} \). We have two cases:

A. If \( i = i \), then by the induction hypothesis, \( \pi(s_i) \geq \pi(s'_i) \) and, by definition of \( \mu_\pi, i \in \mu_\pi(f) \).

B. If \( i \neq i \), then \( s_i = s'_i \) and we have \( \pi(s_i) = \pi(s'_i) \).

Note that \( \pi(s) \) is the \( j \)th immediate subterm of \( \pi(s) \). By \( \mu_\pi \cdot \text{monotonicity of } \preceq \),

\[
\pi(s) = \pi(f(s_1, \ldots, s_k)) = f(\pi(s_1), \ldots, \pi(s_k)) \\
\geq f(\pi(s'_1), \ldots, \pi(s'_k)) = \pi(f(s'_1, \ldots, s'_k)) = \pi(s')
\]

where we assume that \( i = i \) for some \( j \in \{1, \ldots, k\} \). If no such \( j \) exists, then we would have \( \pi(s) = \pi(s') \), which also implies \( \pi(s) \geq \pi(s') \) because \( \preceq \) is reflexive.

Thus, we have proved that \( s \rightarrow s' \) implies \( \pi(s) \geq \pi(s') \) as desired. Therefore, \( \pi(s) \geq \pi(s') \) and, by the induction hypothesis, \( \pi(f) \geq \pi(f) \). By transitivity of \( \preceq \), we conclude \( \pi(s) \geq \pi(t) \).

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 \( R = (F, \pi) \), we write \( \pi(R) \) to denote the filtered TRS \( (F, \pi(R)) \).

10.2. Removing pairs using \( \mu \)-reduction pairs

Given TRSs \( R = (F, \pi) \) and \( P = (G, \mu) \), we can check the absence of infinite minimal \( (P, \pi, \mu) \)-chains often be 'simplified' to checking the absence of infinite minimal \( (P', \pi, \mu) \)-chains for a proper subTRS \( P' \) of \( P \) by finding appropriate \( \mu \)-reduction pairs \( \preceq \).

We need to ensure that the quasi-ordering \( \preceq \) is able to 'look' for a \( \mu \)-replacing subterm \( s \) in the instantiation \( \pi(x) \) of a migrating variable \( x \); since \( \pi(x) = s[l* \, r*] \) for some context \( C[l* \, r*] \) and \( \mu \)-replacing position \( p \in \text{Proj}(C[l* \, r*]) \) such that \( \text{Proj}(C[l* \, r*]) \subseteq C[l* \, r*] \), we can obtain \( s \) out from \( s[l* \, r*] \) by applying the projection rules in \( \text{Emb}^0(f) \) (Definition 1).

Hence, we require \( \text{Emb}^0(f) \subseteq \preceq \).

2. We need to connect the marked version \( s \) (which is an instance of a hidden term \( t \in N^{\pi}(F) \), i.e., \( s = \sigma(t) \) for some substitution \( \sigma \) with an instance \( \pi(x) \) of the left-hand side \( u \) of a pair; hence, we require \( \text{Emb}^0(f) \subseteq \preceq(T) \) for all \( t \in N^{\pi}(F) \) which, by stability, becomes \( s \geq s \) or \( s \geq s \).

The following theorem formalizes a generic process to remove pairs from \( P \) by using argument filterings and \( \mu \)-reduction pairs.

Theorem 9 (\( \mu \)-reduction pair processor).

Let \( R = (F, \pi) \) and \( P = (G, \mu) \) be TRSs and \( \mu \in M_{P,F,G} \). Let \( \pi \) be an argument filtering for \( F \cup G \) and \( \preceq \) is a \( \mu_\pi \)-reduction pair such that

1. \( \pi(R) \preceq \pi(P) \preceq \pi(G) \preceq \pi(F) \)
2. whenever \( N^{\pi}(F) \neq \emptyset \) and \( P \neq \emptyset \), \( \text{we have that}
3. for all \( f \in F \), either \( \pi(f) = \{i_1, \ldots, i_m\} \) and \( \mu(f) \subseteq \pi(f) \), or \( \pi(f) = \{i\} \) and \( \mu(f) = \{i\} \).
is an infinite minimal $(\mu, \mu)$-sequence that contradicts the well-foundedness of $\mu$.

For some substitution $\theta$, where all pairs in $\Xi$ are infinitely often used. Also, for all $i \geq 1$, (1) if $u_i \rightarrow v_i \in \Xi$, then $t_i = \sigma(v_i)$ and (2) if $u_i \rightarrow v_i = u_i \rightarrow x_i \in \Xi$, then $t_i = s_i^v$ for some $s_i$ such that $\sigma(s_i) = C_i[s_i]_{\mu}$ for some $C_i$ and $p_i \in \text{Pos}(\{C_i\})$ such that $\text{spec}(C_i[s_i]) \subseteq F \text{ and } s_i = \theta_i$ for some $\theta_i \in \text{NHT} F$ and substitution $\theta_i$. Actually, since $t_i = s_i^v = \theta_i(s_i^v)$ and $t_i \rightarrow_{\sigma, \mu} \sigma(u_i)$, we can further say that $\hat{u}_i \in \text{NHT} F$.

Since $\pi(u_i)(\geq \cup \supset) \pi(v_i)$ for all $u_i \rightarrow v_i \in \Xi \subseteq P$, by stability of $\geq$ and $\supset$, we have $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(v_i))$ for all $i \geq 1$.

No pair $u \rightarrow v \in \Xi$ satisfies that $\pi(u) \supset \pi(v)$. Otherwise, we get a contradiction by considering the following two cases:

1. If $u_i \rightarrow v_i \in \Xi$, then $t_i = \sigma(v_i) \rightarrow_{\sigma, \mu} s_i^v(\mu)$ and by Proposition 12, $\pi(t_i) \supset \pi(\sigma(u_i))$. By Lemma 7, $\pi(t_i) \supset \sigma_\pi(\pi(u_i))$. Since we have $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$ (using Lemma 7), by using transitivity of $\supset$ and compatibility between $\geq$ and $\supset$, we conclude that $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(u_i))$.

2. If $u_i \rightarrow v_i = u_i \rightarrow x_i \in \Xi$, then $\sigma(v_i) = \sigma(x_i) = C_i[s_i]_{\mu}$.

Since $i \in \mu(f)$, we have that $\pi(x_i) = \pi(\theta_i)$ (using Lemma 7 again) and, similarly, $\pi(t_i) = \theta_i$. By stability we have that $\pi(t_i) \supset \pi(\theta_i)$. Hence, by transitivity of $\supset$ and compatibility between $\geq$ and $\supset$, we have $\sigma_\pi(\pi(v_i)) = \sigma_\pi(\pi(t_i))$ for all $i \in \text{NHT} F$, $\hat{u}_i \in \mu$, and, similarly, $\sigma_\pi(\pi(x_i)) = \sigma_\pi(\pi(t_i))$. Therefore, by transitivity of $\supset$ and compatibility between $\geq$ and $\supset$, we conclude that $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(u_i))$.

Since $u \rightarrow v$ occurs infinitely often in $\hat{B}$, there is an infinite set $I \subseteq \Xi$ such that $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(u_i))$ for all $i \in I$.

And we have $\sigma_\pi(\pi(u_i))(\geq \cup \supset) \sigma_\pi(\pi(u_i))$ for all other $u_i \rightarrow v_i \in \Xi$. Thus, by using the compatibility conditions of the $\mu_\pi$-reduction pair, we obtain an infinite decreasing $\supset$-sequence that contradicts the well-foundedness of $\supset$.

Therefore, $\Xi \subseteq P \rightarrow^{P \rightarrow}$, which means that $\hat{B}$ is an infinite minimal $(P \rightarrow P, R, \mu)$-chain, thus leading to a contradiction. □

**Example 18.** Consider the TRS $R$ [63, Example 5]:

$\begin{align*}
1 &\iff \text{true}, \ x, y \rightarrow x \\
2 &\iff \text{false}, \ x, y \rightarrow y
\end{align*}$

with $\mu(1) = \{1\}$ and $\mu(2) = \{1, 2\}$. Then, $\text{DP}(R, \mu)$ consists of the following CSDPs:

$\begin{align*}
1 &\iff \text{false}, \ x, y, z \rightarrow y
\end{align*}$

with $\mu(1) = \{1\}$ and $\mu(2) = \{1, 2\}$. The $\mu$-reduction pair $(\geq, >)$ induced by the polynomial interpretation$^6$

$\begin{align*}
&[c] = [\text{true}] = 0 \\
&[t](x) = x \\
&[f](x) = x \\
&\text{false} = 1 \\
&\text{true}(x, y, z) = x + y + z \\
&\text{false}(x, y, z) = x + z
\end{align*}$

---

$^6$ See [49] for more information about the automatic generation of polynomial (quasi-)orderings with monotonicity requirements specified by means of replacement maps.
can be used to prove the $\mu$-termination of $R$. For $P = DP(R, \mu)$, we have $NHT_P = \{ t(\text{true}) \}$. First, we can see that the quasi-ordering is compatible with the rules in $\mathcal{EM}_P^\mu(F)$:

\[
\begin{align*}
[t(a)] & = x \quad \triangleright x = [x] \\
[l(t(x, y, z)] & = x + y + z \triangleright x = [x] \\
[l(t(x, y, z)] & = x + y + z \triangleright y = [y]
\end{align*}
\]

Now we can see that the condition on the only hidden term in $NHT_P$ is also fulfilled:

\[
[t(\text{true})] = 0 \triangleright 0 = [P(\text{true})]
\]

Finally, for the three rules in $R$ and the two pairs in $P$, we have:

\[
\begin{align*}
[t(a)] & = x \quad \triangleright x = [l(t(x, c, t(\text{true})))] \\
[l(t(\text{true}, x, y)] & = x + y \triangleright x = [x] \\
[l(t(\text{true}, x, y)] & = x + y + 1 \triangleright y = [y] \\
[P(x)] & = x \quad \triangleright x = [l(t(x, c, t(\text{true})))] \\
[lP(t(\text{false}, x, y)] & = y + 1 \triangleright y = [y]
\end{align*}
\]

We remove the ‘strict’ pair $IF(t(\text{false}, x, y]) \rightarrow y$ from $P$ to obtain $P'$. With $(P', R, \mu^g)$, the application of ProcSCC leads to an empty set of CS problems. Thus, the $\mu$-termination of $R$ 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 $R$:

\[
a \rightarrow c(h(t(a), b))
\]

\[
t(c(x)) \rightarrow x
\]

together with the replacement map $\mu$ given by $\mu(a) = \mu(h) = [1]$ and $\mu(c) = \emptyset$. $DP(R, \mu)$ is:

\[
\begin{align*}
f(\text{true}) & \rightarrow \mu^g(c(h(t(a), b))) \\
f(\text{true}) & \rightarrow \mu^g(h(t(a), b)) \rightarrow \cdots
\end{align*}
\]

and $NHT_{DP(R, \mu)} = \{ t(a) \}$. Note that $R$ is not $\mu$-terminating:

\[
t(t(a)) \rightarrow \mu(c(h(t(a), b))) \\
t(t(a)) \rightarrow \mu(h(t(a), b)) \rightarrow \cdots
\]

For the argument filtering $\pi$ given by $\pi(a) = \pi(h) = [1], \pi(f) = \pi(t) = [1]$ and $\pi(c) = 1, F_g$ consists of the constants $a, h$ and symbol $t$ of arity 1. Also, $\mu^g_1(t) = \mu^g_2(t) = [1]$ and $\mu^g_1(a) = \mu^g_2(h) = \emptyset$. We get the constraints:

\[
\begin{align*}
\pi(a) & = a \triangleright h = \pi(c(h(t(a), b))) \\
\pi(t(c(x))) & = t(x) \triangleright x = \pi(x) \\
\pi(t(a)) & = t(a) \triangleright v(a) = \pi(v(a)) \\
\pi(t(c(x))) & = t(x) \triangleright x = \pi(x)
\end{align*}
\]

which are easily satisfiable (by a polynomial interpretation, for instance). We would wrongly conclude $\mu$-termination of $R$. Note that $\pi(a) = 1$ but $\mu^g_1(a) = \emptyset$, and that $\pi(h) = [1]$ but $\mu^g_2(h) = [1]$.

The next processor is useful when all (filterings of) terms in $NHT_P$ are ground. The advantage is that the quasi-ordering $\triangleright$ of the $\mu$-reduction pair does not need to impose compatibility with the rules in $\mathcal{EM}_P^\mu(F)$.

**Theorem 10 (\mu-reduction pair processor for ground hidden terms).** Let $R = (F, R)$ and $\mu = (\mu, P)$ be TRSs and $\mu \in M_{\mathcal{EM}_P^\mu}$. Let $\pi$ be an argument filtering for $F \cup \triangleright \emptyset$ such that, for all $t \in NHT_P$, $\pi(t)$ is ground. Let $\mu_\triangleright \triangleright$ be a $\mu_\triangleright$-reduction pair such that
Consider the TRS and Lemma 7, we have that $u$ Theorem 10 can succeed when Theorem 9 fails.

Let $\mathcal{P}_x = \{ u \to v \in \mathcal{P}_x \mid \pi(u) \not\sqsubseteq \sqsubseteq \pi(v) \} \cup \{ u \to v \in \mathcal{P}_x \mid \forall t \in \mathcal{NNT}_F, \pi(u) \not\sqsubseteq \sqsubseteq \pi(v) \}. \) Then, the processor $Proc_{\mathcal{P}_x}$ given by

$\begin{align*}
Proc_{\mathcal{P}_x}(\mathcal{P}, \mathcal{R}, \mu) = \begin{cases} 
(\mathcal{P}, R, \mu) & \text{if (1) and (2) hold} \\
(\mathcal{P}, R, \mu) & \text{otherwise}
\end{cases}
\end{align*}$

is sound and complete.

**Proof.** The proof is analogous to that of Theorem 9. Assume the facts and notation in the first paragraph of such a proof. Again, we proceed by contradiction and assume that a pair $u \to v$ is in $\mathcal{P}_x$. Again, we have $\sigma_x(\pi(u)) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(v))$ for all pairs $u_i \to v_i \in \mathcal{Q}_1$.

Now, if $u_i \to v_i = u_j \to v_j \in \mathcal{Q}_1$, then since $\pi(u_i) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(v_j)$ for all $t \in \mathcal{NNT}_F \subseteq \mathcal{NNT}_F$ by stability we have that $\sigma_x(\pi(u_i)) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(v_j))$. Since $t$ is ground, we have $\sigma_x(\pi(u_i)) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(v_j))$. Therefore, since $x_i \in \mathcal{NNT}_F$ and $t_i = t_j$, we have $\sigma_x(\pi(u_i)) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(t_j))$. Finally, since $x_i = x_j$ and $u_i \to v_i \in \mathcal{P}_x$ for all $i \geq 1$, by Proposition 12 and Lemma 7, we have that $\pi(t_i) \not\sqsubseteq \sqsubseteq \pi(u_i)$ (1) and (2) hold. Thus, we also have $\sigma_x(\pi(u_i)) (\not\sqsubseteq \sqsubseteq \not\sqsubseteq \sqsubseteq \pi(u_i)) (1)$ and (2) hold.

Since $u \to v$ occurs infinitely often in $\mathcal{B}$, by using the compatibility conditions of the $\mu$-reduction pair, we obtain an infinite decreasing $\not\sqsubseteq$-sequence that contradicts well-foundedness of $\sqsubseteq$. In particular, if $u \to v \in \mathcal{Q}_1 \cap \mathcal{P}_x$, then $\pi(u) \not\sqsubseteq \sqsubseteq \pi(v)$ for all $t \in \mathcal{NNT}_F$, so each time that $u \to v$ is used, a strict decrease occurs. □

Theorem 10 can succeed when Theorem 9 fails.

**Example 20.** Consider the TRS $R$: $a \to t(d(c(a)))$ (27)

$t(c(x)) \to x$ (28)

$d(c(x)) \to b$ (29)

and the replacement map $\mu$ given by $\mu(c) = \emptyset$ and $\mu(t) = \mu(a) = \{1\}$. There are three CSDPs:

$\lambda \to t(d(c(a)))$ (30)

$\lambda \to d(c(a))$ (31)

$F(c(x)) \to x$ (32)

$Proc_{\mathcal{P}_x}(DP(R, \mu), \mathcal{R}, \mu)$ yields a single CSp problem $(\mathcal{P}, \mathcal{R}, \mu)$ with $\mathcal{P} = \{(30), (32)\}$. Since $\mathcal{NNT}_F = \{a\} \not\sqsubseteq \emptyset$ and $F(c(x)) \to x$ is a collapsing CSDP, according to Theorem 9 we would require that any $\mu$-reduction ordering used in the theorem satisfy $\lambda \not\sqsubseteq \emptyset$ (\emptyset \sqsubseteq \lambda) and that $\lambda \not\sqsubseteq \emptyset \sqsubseteq \lambda$. In this case, though, since $d(c(a)) \sqsubseteq \lambda$, we must have $d(c(a)) \sqsubseteq \lambda(a)$ by $\lambda$-monotonicity of $\lambda$. Thus, one of the following

1. $\lambda \sqsupseteq F(d(c(a)))$ and $F(c(x)) \sqsubseteq \lambda x$. By stability of $\sqsubseteq$ and $\sqsubseteq$, we have $F(c(a)) \sqsubseteq \lambda x$. Thus,

$\lambda \sqsupseteq F(d(c(a))) \sqsupseteq F(c(x)) \sqsubseteq \lambda x$.

By compatibility of $\sqsupseteq$ and $\sqsubseteq$, we have $\lambda \sqsupseteq \lambda \sqsubseteq \lambda \sqsupseteq \lambda \sqsubseteq \lambda$ contradicting the well-foundedness of $\sqsubseteq$.

2. $\lambda \sqsupseteq \lambda \sqsubseteq \lambda$ and $\square \sqsubseteq \lambda \sqsupseteq F(c(a)))$ and $F(c(x)) \sqsubseteq x$. Hence,

$\lambda \sqsupseteq F(d(c(a))) \sqsubseteq F(c(a))) \sqsubseteq \lambda x \sqsubseteq \lambda x$.

Again, by compatibility of $\sqsupseteq$ and $\sqsubseteq$, we have $\lambda \sqsupseteq \lambda \sqsubseteq \lambda \sqsupseteq \lambda \sqsubseteq \lambda$. Thus, Theorem 9 cannot be used with this example. Since $\mathcal{NNT}_F \sqsubseteq T(F)$, Theorem 10 is applicable here. The $\mu$-reduction pair $(\lambda, \rightarrow)$ is induced by the following polynomial interpretation: $^7$

$^7$ See [49, 51] for details about the use of polynomial interpretations with rational coefficients.
Example 21. Consider $R$ 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 $\sqsubseteq$ for any strict component $\langle \cdot, \cdot \rangle$ of any $\mu$-reduction pair $\langle \cdot, \cdot \rangle$. However, the $\mu$-termination of $R$ can be easily proved with Theorem 9. The $\mu$-reduction pair $\langle \cdot, \cdot \rangle$ generated by the following polynomial interpretation:

\[
\begin{align*}
[a] &= 1 \\
[c](x) &= x + 1 \\
[f](x) &= 0
\end{align*}
\]

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 CSSPs 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 $NHTP$, then we can remove all collapsing pairs at the same time.

**Theorem 11** ($\mu$-reduction pair processor for collapsing pairs). Let $R = (F, R, \mu)$ and $P = (G, P, \mu)$ be TRSs and $\mu \in M_{P,G}$. Let $\pi$ be an argument filtering for $F \cup G$ and $\langle \cdot, \cdot \rangle$ be a $\mu$-reduction pair such that

1. $\pi(R) \subseteq \pi(P) \subseteq \bigcup \subseteq$,
2. $\pi(t) \subseteq \pi(f)$ for all $t \in NHTP$ and for all $f \in F$, either $\pi(f) = \{i_1, \ldots, i_n\}$ and $\mu(f) \subseteq \pi(f)$, or $\pi(f) = i$ and $\mu(f) = [i]$.
3. $\exists Y^{pb} \subseteq Y$.

Then, the processor $\text{Proc}_{\text{CSSP}}$ given by

\[
\text{Proc}_{\text{CSSP}}(P, R, \mu) = \begin{cases} 
\{P, R, \mu\} & \text{if (1) and (2) hold} \\
\{P, R, \mu\} & \text{otherwise}
\end{cases}
\]

is sound and complete.

**Proof.** As in the proof of Theorem 9, we proceed by contradiction. We assume that there is an infinite minimal $\langle P, R, \mu \rangle$-chain $A$, but that there is no infinite minimal $(P, R, \mu)$-chain. Thus, there is $Q \subseteq P$ such that $Q \cap P \neq \emptyset$ and $A$ has a tail $B$ as in the proof of Theorem 8. Now, we assume the notation as in the first paragraph of such a proof.

We have $\pi(Q) \subseteq Y$ for all $t \in NHTP$, we now have that $\sigma(t)(Q) \subseteq \pi(t)(Q)$ for all pairs $u_j \rightarrow v_j \in Q$. If $u_j \rightarrow v_j = u_i \rightarrow x_i \in Q$, then by applying the considerations in the corresponding item of the proof of Theorem 9 and taking into account that $\sigma(t)(Q)$ for all $t \in NHTP$, we now have that $\sigma(t)(Q) \subseteq Y$ for all pairs $u_j \rightarrow v_j \in Q$. If $u_i \rightarrow v_i \in Q$, occur infinitely often in $B$, by using the compatibility conditions of the $\mu$-reduction pair, we obtain an infinite decreasing $\sqsubseteq$-sequence that contradicts the well-foundedness of $\sqsubseteq$. □
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 \( \mathcal{R} = (F, R) \) be a TRS. The set of root symbols associated to \( \mathcal{R} \) is:

\[
\text{Root}(\mathcal{R}) = \{ \text{root}(f) \mid f \in F \} \cup \{ \text{root}(r) \mid r \in R, r \notin X \}
\]

The following result relates \( \text{Root}(\mathcal{R}) \) and the set \( \mathcal{N}_F \) of hidden symbols occurring at the root of terms in \( \mathcal{N}^F \mathcal{T}_F (\mathcal{R}, \mu) \). It is silently used in the statements of some theorems below.

**Lemma 8.** Let \( \mathcal{R} = (F, R) = (\mathcal{C} \cup \mathcal{D}, R) \) and \( \mathcal{P} = (\mathcal{Q}, P) \) be TRSs such that \( \text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset \), and \( \mu \in M_{\mathcal{F}, \mathcal{D}}. \) For all \( f \in \mathcal{N}_P \), we have \( f \notin \text{Root}(\mathcal{P}) \).

**Proof.** If \( f \in \mathcal{N}_P \), then there is \( t \in \mathcal{N}^F \mathcal{T}_F (\mathcal{R}, \mu) \) such that \( f = \text{root}(t) \). Therefore, there are substitutions \( \theta \) and \( \theta' \) such that \( \theta(t') \overset{\mathcal{P}}{\rightarrow} \theta'(u) \) for some \( u \rightarrow v \in P \). Since \( f \notin \mathcal{F}, \mu \)-rewritings on \( \mathcal{R} \) do not remove it. Thus, \( \text{root}(u) = f \) and \( f \in \text{Root}(\mathcal{P}) \). \( \square \)

Thiemann uses argument filtrations (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 \( i \in \{1, \ldots, k\} \). The mapping that assigns a subterm \( \pi(t) = t_{\langle i \rangle} \) to every term \( t = f(t_1, \ldots, t_k) \) with \( f \in \text{Root}(\mathcal{R}) \) is also denoted by \( \pi; \) we also let \( \pi(x) = x \) if \( x \in X \).

Given a simple projection \( \pi \) for a TRS \( \mathcal{R} \), we let \( \pi(\mathcal{R}) = [\pi(I) \rightarrow \pi(t) \mid I \rightarrow t \in \mathcal{R}] \).

**Theorem 12** (Subterm processor for noncollapsing pairs). Let \( \mathcal{R} = (F, R) = (\mathcal{C} \cup \mathcal{D}, R) \) and \( \mathcal{P} = (\mathcal{Q}, P) \) be TRSs such that \( \mathcal{P} \) contains no collapsing rule, i.e., all \( u \rightarrow v \in P, v \notin X, \) and \( \text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset \). Let \( \mu \in M_{\mathcal{F}, \mathcal{D}} \) and let \( \pi \) be a simple projection for \( \mathcal{P} \). Let \( \mathcal{P}_{\pi, \mathcal{D} \times \mu} = \{ u \rightarrow v \in P \mid \pi(u) \supseteq \pi(v) \} \). Then, the processor \( \text{Proc}_{\text{subNColl}} \) is given by

\[
\text{Proc}_{\text{subNColl}}(\mathcal{P}, \mathcal{R}, \mu) = \left\{ \begin{array}{ll}
(\mathcal{P} \setminus \mathcal{P}_{\pi, \mathcal{D} \times \mu}, \mathcal{R}, \mu) & \text{if } \mathcal{P}_{\pi, \mathcal{D} \times \mu} \subseteq \mathcal{P} \\
(\mathcal{P}, \mathcal{R}, \mu) & \text{otherwise}
\end{array} \right.
\]

is sound and complete.

**Proof.** Completeness is obvious because \( \mathcal{P} \setminus \mathcal{P}_{\pi, \mathcal{D} \times \mu} \subseteq \mathcal{P} \). 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 \( (\mathcal{P} \setminus \mathcal{P}_{\pi, \mathcal{D} \times \mu}, \mathcal{R}, \mu) \)-chain. Since \( \mathcal{P} \) is finite, we can assume that there is \( \mathcal{Q} \subseteq \mathcal{P} \) such that \( A \) has a tail \( \mathcal{B} \) that is an infinite minimal \((\mathcal{Q}, \mathcal{R}, \mu)\)-chain where all pairs in \( \mathcal{Q} \) are infinitely often used. Assume that \( \mathcal{B} \) is as follows (since \( \mathcal{Q} \subseteq \mathcal{D} \), we use a simpler notation):

\[
t_0 \overset{\mathcal{A}}{\rightarrow}_{\mathcal{Q}, \mathcal{P}} t_1 \overset{\mathcal{A}}{\rightarrow}_{\mathcal{Q}, \mathcal{P}} t_2 \overset{\mathcal{A}}{\rightarrow}_{\mathcal{Q}, \mathcal{P}} \cdots
\]

where there is a substitution \( \sigma \) such that, for all \( i \geq 1, s_i = \sigma(t_i) \) 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 \( v_0 \rightarrow v_0 \in \mathcal{P} \).

Note that, for all \( i \geq 1, \text{root}(s_i) \in \text{Root}(\mathcal{P}) \). Hence, \( \text{root}(v_0) \notin \mathcal{D} \) and we can actually write \( t_i \overset{\mathcal{A}}{\rightarrow}_{\mathcal{Q}, \mathcal{P}} s_{i+1} \) because \( \mathcal{P} \)-rewritings with \( \mathcal{K} \) cannot change \( \text{root}(t_i) \). Hence, \( \pi(t_i) \overset{\mathcal{P}}{\rightarrow}_{\mathcal{Q}, \mathcal{P}} \pi(s_{i+1}) \) and also \( \text{root}(t_i) = \text{root}(s_{i+1}) \) for all \( i \geq 0 \). Finally, since \( \pi(v_0) \supseteq \pi(v_i) \) for all \( i \geq 0 \), by stability of \( \mathcal{D}, \mu \), we have

\[
\pi(s_i) = \pi(\sigma(t_i)) = \sigma(\pi(t_i)) \supseteq \sigma(\pi(v_i)) = \pi(\sigma(v_i)) = \pi(\pi(v_i)) = \pi(t_i)
\]

for all \( i \geq 1 \). No pair \( u \rightarrow v \in \mathcal{Q} \) satisfies that \( \pi(u) \supseteq \pi(v) \). Otherwise, we get a contradiction in both of the following two complementary cases:
Example 22 (Proof of termination of the main example). Consider the termination problems obtained in Example 15 for the CS-TRS in Example 1:

\[ \tau_1 = (\{(1\}) , R, \mu^1) , \quad \tau_2 = (\{(17\}) , R, \mu^1) , \quad \tau_3 = (\{(21\}) , R, \mu^1) , \quad \text{and} \quad \tau_4 = (\{(23\}) , R, \mu^1) \]

We apply \( \text{Proc}_\text{subNColl} \) to all such problems. For \( \tau_1 \) with \( \pi(\text{ADD}) = 1 \), we have \( \pi(\text{ADD}(a(n), m)) = a(n) \triangleright \mu^n = \pi(\text{ADD}(a(n), m)) \). Now, \( \text{Proc}_\text{subNColl}(\tau_1) = \{ (\emptyset, R, \mu^2) \} \). With \( \text{Proc}_\text{subNColl} \) we conclude that \( \tau_1 \) is finite. Since this can be done for \( \tau_2, \tau_3, \) and \( \tau_4 \), the \( \mu \)-termination of \( S \) is proved.

The following examples show that if \( P \) contains collapsing rules, then Theorem 12 does not hold.

Example 23. Consider the two TRSs

\[ R : h(x) \rightarrow f(g(h(x))) \quad \text{and} \quad P : f(x) \rightarrow x \]

Let \( \mu \) be given by \( \mu(f) = \{ 1, \ldots, k \} \) for all symbols \( f \). Note that, \( \text{Root}(P) = \{ f \} \) and \( D = \{ x \} \) are disjoint. By using the projection \( \pi(t) = 1 \), we get \( \pi(f(g(h(x)))) = g(x) \triangleright \mu^n x. \) After removing the pair in \( P \), a finite CS problem \( (\emptyset, R, \mu) \) is obtained. However, \( (P, R, \mu) \) is not finite:

\[ f(g(h(x))) \rightarrow_{P,R} h(x) \rightarrow_{R} g(x) \rightarrow_{R} \ldots \]

In the following theorem, we show how to use the subterm criterion to remove all collapsing pairs from \( P \).

Theorem 13 (Subterm processor for collapsing pairs). Let \( R = (f, R) = (c \cup D, R) \) and \( P = (f, P) \) be TRSs such that \( P \) contains no collapsing rule, \( \text{Root}(P) \cap D = \emptyset \), and \( \mu \in M_{P,R} \). Let \( \pi \) be a simple projection for \( P \) such that

1. \( \pi(f) \subseteq P, \) and
2. whenever \( P, \emptyset \) is disjoint, we have \( \pi(f^2) \in P, \emptyset \cap \mu(f) \) for all \( f \in N_P \).

Then, the processor \( \text{Proc}_\text{subColl} \) given by

\[ \text{Proc}_{\text{subColl}}(P, R, \mu) = \left\{ \begin{array}{ll}
\{(P, R, \mu)\} & \text{if (1) and (2) hold} \\
\{(P, R, \mu)\} & \text{otherwise}
\end{array} \right. \]

is sound and complete.

Proof. Completeness is obvious because \( P \subseteq P \). For soundness, we proceed by contradiction. Assume that there is an infinite minimal \( (P, R, \mu) \)-chain \( A \) but there is no infinite minimal \( (P, R, \mu) \)-chain. Since \( P \) is finite, we can assume that
there is \( Q \subseteq P \) such that \( A \) has a tail \( B \) which is an infinite minimal \((Q, R, \mu)\)-chain where all pairs in \( Q \) are infinitely often used and \( Q \) contains some collapsing pair \( u \rightarrow x \in Q_{\mu} \). Assume that \( B \) is

\[
\begin{align*}
\bar{t}_i &\leftarrow s_{i, \mu} & s_1 &\leftarrow t_1 & t_1 &\leftarrow t_2 & t_2 &\leftarrow t_3 & \cdots
\end{align*}
\]

where there is a substitution \( \sigma \) such that, for all \( i \geq 1 \), \( s_i = \sigma(u_i) \) for some \( u_i \rightarrow v_i \in P \), and

1. if \( v_i \notin \Lambda \), then \( t_i = \sigma(v_i) \),
2. if \( v_i = x_i \in \Lambda \), then \( u_i \not\in \var N(u_i) \) and \( t_i = t_i^u \) for some \( t_i \in T(f, x) \) such that \( \sigma(v_i) = \sigma(t_i^u) \) for some \( \sigma \in N_{[b]} \) and \( t_i \in \var N(t_i^u) \) such that \( \operatorname{prefix}(t_i^u) \subseteq \var N(t_i^u) \) and substitution \( \sigma \).

Since we can freely choose the starting term of \( B \), w.l.o.g. we assume that \( B \) is a particular case of the second alternative above, i.e., there is a collapsing pair \( u_0 \rightarrow x_0 \) such that \( \sigma(u_0) \subseteq P \) and \( t_0 = t_0^u \). Note that, for all \( i \geq 1 \), \( \sigma(u_i) \in \var N(P) \) because \( \sigma(u_i) \) is a root of \( P \). Furthermore, for all \( i \geq 0 \), \( \sigma(u_i) \in \var N(P) \) because:

1. If \( u_i \rightarrow v_i \in Q_{\mu} \), then root \( (v_i) \in \var N(P) \) and \( t_i = \sigma(v_i) \).
2. If \( u_i \rightarrow v_i \in Q_{\mu} \), then root \( (t_i) \in \var N(P) \) since \( t_i \subseteq s_i+1 \) and \( \var N(t_i) = \var N(x) \) and \( \var N(x) \) occurs infinitely often in \( B \). Therefore, for all \( i \geq 0 \), we can write \( t_i \leftarrow s_i+1 \). Hence, \( \var N(t_i) = \var N(s_i) \) and also root \( (t_i) \) is root \( (s_i+1) \) for all \( i \geq 0 \).

Since \( u \rightarrow x \in \var N(P) \) and \( v \) is infinite, it must be \( \var N(P) \not\var N(P) \) (here \( \var N(P) \not\var N(P) \) ). Thus, we have \( \var N(f) \in \mu(f) \) for all \( f \in \var N(P) \). Then, since root \( (t_0) = \var N(s_0) \) and all pairs in \( Q \) occur infinitely often in \( B \), we can assume that \( \var N(t_0) = \var N(t) \). Furthermore, since \( A \) is minimal, we can assume that \( t_0 \) is \( \mu \)-terminating. We have that \( \sigma(u_0) \subseteq P \) and \( \sigma(u_0) \) for all \( v_i \rightarrow u_i \in Q_{\mu} \). Now we distinguish two cases:

1. If \( u_i \rightarrow v_i \in Q_{\mu} \), then \( s_i = \sigma(u_i) \) and \( t_i = \sigma(v_i) \). By stability of \( \var N(P) \) we have \( \var N(s_i) \subseteq P \). By case (2) above, \( B \) contains infinitely many \( \var N(s) \)-steps, starting from \( \var N(t_0) \). Since \( \var N(s) \) is well-founded, the infinite sequence must also contain infinitely many \( \var N(s) \)-steps. By making repeated use of the fact that \( \var N(P) \subseteq \var N(P) \subseteq \var N(P) \), we obtain an infinite \( \var N(P) \)-sequence starting from \( \var N(t_0) \). Thus, \( \var N(t_0) \) is not \( \mu \)-terminating with respect to \( P \). Since \( \var N(f) \in \mu(f) \) and hence \( \var N(f) \), this implies that \( \var N(t_0) \) is not \( \mu \)-terminating (use Lemma 1). This contradicts \( \mu \)-termination of \( t_0 \). Therefore, \( Q \) cannot contain collapsing pairs. This contradicts our initial assumption \( u \rightarrow x \in Q_{\mu} \).

Remark 13. The use of Theorem 13 only makes sense if \( P \subseteq P \cup P \). If \( u \rightarrow x \in P \) for some \( u = f(u_1, \ldots, u_k) \), then for all \( i \in \{1, \ldots, k\} \), whenever \( x = \var N(u_i) \) we have \( \mu(f) \) and \( u_i \). Thus, there is no simple projection \( P \) such that \( \var N(u_i) \subseteq P \).

Example 24. Consider the following TRS \( \mathcal{R} \):

\[
\begin{align*}
q(x, y) &\rightarrow \varepsilon(x, y) \\
\varepsilon(\varepsilon(x, y)) &\rightarrow q(x, q(y, y))
\end{align*}
\]

together with the replacement map \( \mu \) given by \( \mu(q) = [1] \) and \( \mu(\varepsilon) = \varnothing \). The CSDPs are:
Theorem 14. Let $\pi(\sigma)$ be a well-founded and $\mu$-replacing projection. Then, $\pi(\sigma)$ is sound and complete.

Proof. We have that $\forall R \subseteq \mathcal{P}$ such that

1. for all $f \in \text{Root}(\mathcal{P})$, $\pi(f) \neq \mu(f)$,
2. $\pi(\mathcal{P}) \subseteq \mathcal{R}$.
3. whenever $\mathcal{N^T}_R \neq \emptyset$ and $\mathcal{P}_R \neq \emptyset$, we have that $\mathcal{N^T}_R \subseteq \mathcal{R}$ and $t \geq \mathcal{E}_{\mathcal{N^T}}(p)$ for all $t \in \mathcal{N^T}_R$.

Let $\mathcal{P}_R = \{u \rightarrow v \in \mathcal{P} \mid \pi(u) > \pi(v)\}$. Then, the processor $\mathcal{P}$ given by

$$\mathcal{P}_R = \{(\mathcal{P} - \mathcal{P}_R, \mathcal{R}, \mathcal{\mu}) \text{ if (1), (2), and (3) hold otherwise}\}$$

is sound and complete.

Proof. Completeness is obvious because $\mathcal{P} - \mathcal{P}_R \subseteq \mathcal{P}$. For soundness, we proceed by contradiction. Assume that there is an infinite minimal $(\mathcal{P} - \mathcal{P}_R, \mathcal{R}, \mathcal{\mu})$-chain $A$ but there is no infinite minimal $(\mathcal{P} - \mathcal{P}_R, \mathcal{R}, \mathcal{\mu})$-chain. Since $\mathcal{P}$ is finite, we can assume that there is a $\emptyset \subseteq \mathcal{P}$ such that $A$ has a tail $B$

$$\sigma(u_1) \rightarrow_{\mathcal{R}, \sigma} \sigma(u_2) \rightarrow_{\mathcal{R}, \sigma} \cdots$$

for some substitution $\sigma$ and pairs $u_i \rightarrow v_i \in \mathcal{Q}$, and

1. if $v_i \neq x$, then $u_i = \sigma(u_i)$, and
2. if $v_i = x$, then $v_i \neq \mathcal{N^T}_R(u_i)$ and $t_i = s_i$ for some $s_i$ such that $\sigma(x_i) = C_i[s_i]$, for some $C_i \in \mathcal{C}$, and $\mathcal{P}_R \subseteq \mathcal{P}_R$ such that $\mathcal{N^T}_R(p_i) \subseteq \mathcal{R}$ and $s_i = \sigma_i(t_i)$ for some $s_i \in \mathcal{N^T}_R$, and substitution $\sigma_i$.

Furthermore, all pairs $\sigma$ are used infinitely often in $B$. As discussed in the proof of Theorem 12, for all $i \geq 1$, $\mathcal{t}(u_i) = \sigma(u_i)$ and also $\mathcal{t}(u_i) = \sigma(u_i)$ for all $i \geq 1$. No pair $u \rightarrow v \in \mathcal{Q}$ 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 $\pi(f) \neq \mu(f)$ for all $f \in \text{Root}(\mathcal{Q})$, no $\mathcal{R}$-rewritings are possible on the subterm $\mathcal{t}(u_i)$ of $u_i$. Therefore, for all $i \geq 1$,

2. due to $\pi(u_i) \geq \pi(v_i)$ and by stability of $\mathcal{\mu}$, we have that $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \geq \sigma(\pi(v_i))$. Now, we distinguish two cases:

(a) If $u_i \rightarrow v_i \in \mathcal{Q}$, then $\mathcal{t}(u_i) = \mathcal{t}(v_i) = \pi(\sigma(v_i)) = \sigma(\pi(v_i))$. Thus, $\pi(\sigma(u_i)) \geq \pi(\mathcal{t}(u_i))$.

(b) If $u_i \rightarrow v_i \in \mathcal{Q}$, then $\mathcal{t}(\mathcal{t}(u_i)) = \mathcal{t}(\mathcal{t}(v_i)) = \mathcal{t}(\mathcal{t}(v_i)) = \pi(\mathcal{t}(v_i))$. Since $\mathcal{t}(\mathcal{t}(v_i)) = \pi(\mathcal{t}(v_i))$, we have $\mathcal{t}(\mathcal{t}(v_i)) \geq \pi(\mathcal{t}(v_i))$. Hence, $\pi(\sigma(u_i)) \geq \pi(\mathcal{t}(u_i))$. 

\[\begin{align*}
\sigma(x, y) &\rightarrow \pi(x, y) \\
\pi(x, y) &\rightarrow \omega(x, y) \\
\pi(x, y) &\rightarrow x
\end{align*}\]
Thus, we always have $\pi(\sigma(u_i)) \geq \pi(u_i)$. We obtain an infinite sequence
$$
\pi(\sigma(u_i)) \geq \pi(u_i) = \pi(\sigma(u_j)) \geq \pi(u_j) \cdots
$$
Since pairs in $Q$ occur infinitely often, this sequence contains infinitely many $>$ steps starting from $\pi(\sigma(u_i))$. This contradicts the well-foundedness of $>$. Therefore, $\emptyset \subseteq P - P_m$, i.e., $B$ is an infinite minimal $(P - P_m, R, \mu)$-chain. This contradicts our initial assumption. □

**Example 25.** Consider the CS-TRS $(R, \mu)$ in Example 10. $DP(R, \mu)$ is:

$$
\begin{align*}
\emptyset(x) & \rightarrow \pi(x) \\
\pi(x) & \rightarrow \pi(x)
\end{align*}
$$

where $\mu^*(c) = \mu^*(c) = \emptyset$. The dependency graph contains a single cycle that includes both pairs. The only simple projection is $\pi(\emptyset) = \pi(\emptyset) = 1$. Since $\pi(\emptyset(x)) = \pi(\emptyset(x)) = 1 > c = \pi(\emptyset(c))$ holds for a stable and well-founded ordering $> (e.g., \text{an RPO with } \emptyset > c)$. Theorem 15 (Non-$\mu$-replacing projection processor $\Pi^\mu$). Let $R = (F, R) = (C \cup D, R)$ and $P = (G, P)$ be TRSs such that $P_m$ contains no collapsing rule, $Root(P) \cap \theta \neq \emptyset$, and $\mu \in M_{\text{ps}}$. Let $\Pi\Pi$ be a stable quasi-ordering on terms whose strict and stable part $>$ is well-founded and let $\pi$ be a simple projection for $P$ such that

1. for all $f \in \text{Root}(P)$, $\pi(f) \neq \mu(f)$,
2. $\pi(P) \subseteq \geq$,
3. whenever $\Lambda'N(T, P) \neq \emptyset$ and $P_m \neq \emptyset$, we have that $\emptyset(\sigma(f)) \subseteq \geq$ and $t > t_\emptyset(\sigma(f))$ for all $t \in \Lambda'N(T_P)$.

Then, the processor $Proc_{\text{NARY}}$ given by

$$
\text{Proc}_{\text{NARY}}(P, R, \mu) = \begin{cases} 
\{ (P_m, R, \mu) \} & \text{if } (1), (2), \text{ and } (3) \text{ hold} \\
\{ (P, R, \mu) \} & \text{otherwise}
\end{cases}
$$

is sound and complete.

12. Narrowing transformation

The starting point of a proof of $\mu$-termination of a TRS $R$ is the computation of the estimated CSDG, $\text{EDG}(R, \mu)$, 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]:

$$
\begin{align*}
\sigma(0) & \rightarrow \text{cons}(1, \sigma(a(0))) \\
\sigma(a(0)) & \rightarrow x
\end{align*}
$$

with $\mu(0) = \mu(x) = \mu(\text{cons}) = \{ 1 \}$ and $\mu(\emptyset) = \emptyset$. $DP(R, \mu)$ consists of the pairs:

$$
\begin{align*}
\sigma(a(0)) & \rightarrow \sigma(a(0)) \quad (36) \\
\sigma(a(0)) & \rightarrow \sigma(a(0)) \quad (37)
\end{align*}
$$

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 $\sigma(a(0))$ into $\sigma(a(0))$.

As already observed by Arts and Giesl for the standard case [10], in our case, the overestimation comes when a (noncollapsing) pair $u_i \rightarrow v_i$ 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 $\sigma(v_i) \not\rightarrow_{\mu} \sigma(v_i) \not\rightarrow_{\mu} \sigma(v_i)$. Then, $v_i$ is a one-step $\mu$-narrowing of $v_i$, and we could require $u_i \not\rightarrow v_i$ (which could be easier to prove) instead of $u_i \not\rightarrow v_i$.

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 $v_1, \ldots, v_n$, which are one-step $\mu$-narrowings of $v$ using unifiers $\theta_i$ (i.e., $v \not\rightarrow_{\mu, \theta_i(i)} v_i$) for $i \in \{ 1, \ldots, n \}$, respectively. These unifiers are also applied to the left-hand side of the pair $u \rightarrow v$. Therefore, we can replace a pair $u \rightarrow v$ by all its (one-step) $\mu$-narrowed pairs $\theta_1(u) \rightarrow v_1, \ldots, \theta_n(u) \rightarrow v_n$. 

Publications (full text)
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 \( Q \) is not \( \mu \)-like “no multiple occurrences of the same variable are allowed”.

Example 27. The following TRS is used in [10] to motivate the requirement of linearity.

\[
\begin{align*}
\ell(u(x)) &\to \ell(u(x)) \\
g(u, 1) &\to w(0) \\
0 &\to 1
\end{align*}
\]

We make it a CS-TRS by adding a replacement map \( \mu \) given by \( \mu(\ell) = \mu(w) = \{1\} \), and \( \mu(g) = \{2\} \). The only cycle in the CSDG consists of the CSDP

\[
\bar{\ell}(u(x)) \to \bar{\ell}(g(x, x)).
\]

If linearity of the right-hand sides is not required for \( \mu \)-narrowing CSDPs, then this pair will be removed, since \( \bar{\ell}(g(x, x)) \) and the (renamed version of) the left-hand side \( \bar{\ell}(u(x')) \) do not unify. Thus, there are no \( \mu \)-narrowings. However, the system is not \( \mu \)-terminating:

\[
\ell(u(0)) \leftrightarrow \ell(g(0, 1)) \leftrightarrow \ell(g(0, 1)) \leftrightarrow \ell(u(0)) \ldots
\]

The problem is that the \( \mu \)-reduction from \( \sigma(\bar{\ell}(g(x, x))) \) to \( \sigma(\bar{\ell}(u(x'))) \) takes place “in \( \sigma \)”, and, therefore, it cannot be captured by \( \mu \)-narrowing. Note that \( \bar{\ell}(g(x, x)) \) is “\( \mu \)-linear”.

Another restriction to take into account when \( \mu \)-narrowing a noncollapsing pair \( u \to v \) 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 \( R \) be a TRS and \( \mu \in M\mathcal{E} \). A rule \( I \to r \) is strongly \( \mu \)-conservative if it is \( \mu \)-conservative and \( \text{var}^R(I) \cap \text{var}^R(r) = \emptyset \).

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 \( R \) and \( P \) be TRSs and \( \mu \in M\mathcal{E} \). Let \( u \to v \in P \) be such that

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

Let \( Q = (P - (u \to v)) \cup \{ u' \to v' | u' \to v \text{ is a } \mu \text{-narrowing of } u \to v \} \). Then, the processor \( \text{Proc}_{\text{narrow}} \) given by

\[
\text{Proc}_{\text{narrow}}(P, R, \mu) = \begin{cases} 
\{Q, R, \mu\} & \text{if (1) and (2) hold} \\
\{P, R, \mu\} & \text{otherwise}
\end{cases}
\]

is

1. sound whenever \( u \to v \) is strongly conservative,
2. complete in all cases.

Proof. The proof of this theorem is analogous to the proof of [35, Theorem 31], which we adapt here. For soundness, we prove that given a minimal \( (P, R, \mu) \)-chain \( \ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots \), there is a \( \mu \)-narrowing \( v' \) of \( v \) with the mgu \( \theta \) such that \( v' = \theta(v) \) is also a minimal \( (Q, R, \mu) \)-chain. Hence, every infinite minimal \( (P, R, \mu) \)-chain yields an infinite minimal \( (Q, R, \mu) \)-chain.

If \( \ldots, u_1 \to v_1, u \to v, u_2 \to v_2, \ldots \) is a minimal \( (P, R, \mu) \)-chain, then there is a substitution \( \sigma \) such that for all pairs \( s \to t \) in the chain,

1. if \( s \to t \in \mathcal{P}_C \), then \( \sigma(t) \) is \( \mu \)-terminating and it \( \mu \)-reduces to the instantiated left-hand side \( \sigma(s') \) of the next pair \( s' \to t' \) in the chain,
2. if \( s \to t = s \to x \in \mathcal{P}_C \), then \( \sigma(x) \) has a \( \mu \)-replacing subterm \( \sigma_0(x) \subseteq \mu \), such that \( \sigma_0(x) \) is \( \mu \)-terminating and it \( \mu \)-reduces to the instantiated left-hand side \( \sigma(s') \) of the next pair \( s' \to t' \) in the chain; furthermore, there is \( s_0 \in \mathcal{N}(T(R, \mu)) \) such that \( \sigma_0 = \theta_0(s_0) \) for some substitution \( \theta_0 \).
Assume that $\sigma$ is a substitution satisfying the above requirements and such that the length of the sequence $\sigma(v) \rightarrow_{\mu, p} \sigma(u_2)$ 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 $\sigma(v) \rightarrow_{\mu, q} \sigma(u_2)$. We consider two possible cases:

1. The reduction $\sigma(v) \rightarrow_{\mu, q}$ takes place within a binding of $\sigma$, i.e., there is a term $r$, a $\mu$-replacing variable position $p \in \text{Var}(v)$, and a $\mu$-replacing variable $x \in \text{Var}(v)$ such that $\sigma(v_p) = x \neq \sigma(v)[v_p]$, and $\sigma(x) \rightarrow_{\mu, r}$. Since $v$ is linear, $x$ occurs only once in $v$. Thus, $q = \sigma'(v)$ for the substitution $\sigma'$ with $\sigma'(y) = r$ and $\sigma'(x) = \sigma(x)$ 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 $v \rightarrow v$. We have $\sigma(z) \rightarrow_{\mu, q} \sigma(z)$ for all $z \in X$. Since $v \rightarrow v$ is strongly conservative, we also have $\sigma(u) \rightarrow_{\mu, q} \sigma'(u)$ because all occurrences of $x$ in $u$ must be $\mu$-replacing. Hence, if $u_1 \rightarrow v_1 \in \mathbb{P}$, we have

$$\sigma'(v_1) = \sigma(v_1) \rightarrow_{\mu, q} \sigma(u) \rightarrow_{\mu, q} \sigma'(u),$$

and if $u_1 \rightarrow v_1 \in \mathbb{P}$, then there is $s_1 \in \mathcal{T}(F, X)$ such that

$$\sigma'(v_1) = \sigma(v_1) \rightarrow_{\mu, s_1} \sigma(u) \rightarrow_{\mu, q} \sigma'(u),$$

and, in both cases,

$$\sigma'(v) = q \rightarrow_{\mu, q} \sigma'(u) = \sigma(u_2).$$

Note that, by minimality and because $u \rightarrow v \in \mathbb{P}$, $(\mathbb{R}, \mu)$-terminating and, since $\sigma(v) \rightarrow_{\mu, q}$, the term $q$ is $(\mathbb{R}, \mu)$-terminating as well. Therefore, $\sigma'(x) = q$ is $(\mathbb{R}, \mu)$-terminating and $\sigma'$ satisfies the two conditions above. Since the length of the sequence $\sigma'(v) \rightarrow_{\mu, q} \sigma'(u)$ is shorter than the sequence $\sigma(v) \rightarrow_{\mu, q} \sigma(u)$, we obtain a contradiction and we conclude that the $\mu$-reduction $\sigma(v) \rightarrow_{\mu, q}$ cannot take place in a binding of $\sigma$.

2. The reduction $\sigma(v) \rightarrow_{\mu, q}$ "touches" $v$, i.e., there is a nonvariable position $p \in \text{Var}(v)$, and a rewrite rule $l \rightarrow r \in \mathbb{R}$ such that $\sigma(v_p) = r(l)$ for some substitution $\rho$ and

$$\sigma(v) = (\sigma(v)[v_p])_p = (\sigma(v)[r(l)])_p \rightarrow_{\mu, \rho} \sigma(v)[r(r)]_p = q,$$

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 $\sigma(x) = \theta(\sigma(x))$ for all variables $x$. We have that $v$-$\mu$-narrows to $\theta(\sigma(x))$, behaves like $\sigma(x)$, and if $u_1 \rightarrow v_1 \in \mathbb{P}$, then there is $s_1 \in \mathcal{T}(F, X)$ such that

$$\sigma(v_1) = \sigma(x) \rightarrow_{\mu, s_1} \sigma(u) = \tau(\sigma(u))$$

and

$$\sigma(v') = \tau(v') = \tau(\sigma(x)) \rightarrow_{\mu, s} \sigma(u) = \sigma(v)[\sigma(r)]_p = \sigma(\sigma(r))_p = q \rightarrow_{\mu, q} \sigma(u_2).$$

Hence, "$\ldots, u_1 \rightarrow v_1, \theta(u) \rightarrow v', u_2 \rightarrow v_2, \ldots\"$ is also a minimal chain.

Completeness is also analogous to the 'completeness' part of [35, Theorem 31]. If $(\mathbb{Q}, \mathbb{R}, \mu)$ is infinite and $\mathbb{R}$ is non-$\mu$-terminating, then $(\mathbb{P}, \mathbb{R}, \mu)$ is infinite as well. If $\mathbb{R}$ is $\mu$-terminating, then let "$\ldots, u_1 \rightarrow v_1, \theta(u) \rightarrow v', u_2 \rightarrow v_2, \ldots\"$ be an infinite minimal $(\mathbb{Q}, \mathbb{R}, \mu)$-chain where $v'$ is a one-step $\mu$-narrowing of $v$ using the mgu $\theta$. We prove that "$\ldots, u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, \ldots\"$ is an infinite minimal $(\mathbb{P}, \mathbb{R}, \mu)$-chain. There is a substitution $\sigma$ such that

$$\sigma(v_1) \rightarrow_{\mu, q} \sigma(\theta(u)) \text{ if } u_1 \rightarrow v_1 \in \mathbb{P}$$

and

$$\sigma(v_1) = \sigma(x) \rightarrow_{\mu, s_1} \sigma(u) \rightarrow_{\mu, q} \sigma(u_2) \text{ if } u_1 \rightarrow v_1 \in \mathbb{P}.$$
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,

$$\sigma(v_1) \leftarrow u_{1,\mu} \sigma(u) \quad \text{if } u_1 \rightarrow v_1 \in P^\mu,$$

$$\sigma(v_1) \leftarrow v_{1,\mu} \sigma(u) \quad \text{if } u_1 \rightarrow v_1 \in P^\mu.$$

Moreover, by definition of $\mu$-narrowing, we have $\theta(v) \leftarrow u_{m,\mu} v'$. This implies that $\sigma(\theta(v)) \leftarrow u_{m,\mu} \sigma(v')$, and since $\sigma(v) = \sigma(\theta(v))$, we obtain

$$\sigma(v) \leftarrow u_{m,\mu} \sigma(v').$$

Since $R$ is $\mu$-terminating, $\sigma(v)$ is $(R, \mu)$-terminating. Hence, "... $u_1 \rightarrow v_1, u \rightarrow v, u_2 \rightarrow v_2, ..."$ is a minimal infinite $(P, R, \mu)$-chain as well. □

**Example 28.** Since the right-hand side of pair (36) in Example 26 does not unify with any (renamed) left-hand side of a CSDP (including itself) and it can be $\mu$-narrowed at position 1 (notice that $\mu(\gamma(\theta[1]))$ by using the rule $\theta(\theta(x)) \rightarrow x_c$, we can replace it by its $\mu$-narrowed pair:

$$F(\theta(\theta)) \rightarrow F(\theta).$$

Now, ProcCSCC([36]), $P, R, \mu = \emptyset$ and the $\mu$-termination of $R$ 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 TRS $R$:

$$c(a(x)) \rightarrow d(x, x)$$
$$a \rightarrow c(a)$$

and $P$ consisting of the following pair:

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

together with $\mu(c) = \mu(a) = \mu(f) = \{1\}$ and $\mu(a) = \emptyset$. There is an infinite $(P, R, \mu)$-chain:

$$F(c(d)) \rightarrow F(c(\theta(\theta))) \rightarrow F(c(\theta(\theta(n)))) \rightarrow F(c(\theta(\theta(n)))) \rightarrow F(c(\theta(\theta(n)))) \rightarrow \cdots$$

Since $(\gamma(\theta(\theta(n))))$ does not unify with any left-hand side of another pair, we can $\mu$-narrow the pair in $P$. We obtain $\nu'$ consisting of the $\mu$-narrowed pair:

$$\nu(d(a(x), c(x))) \rightarrow \nu(d(x, x))$$

No infinite $(\nu', R, \mu)$-chain is possible now. Note that $P$ is $\mu$-conservative, but it is not strongly $\mu$-conservative (the variable $x$ is both $\mu$-replacing and non-$\mu$-replacing in $F(d(x, x))$).

**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 $u \rightarrow v \in P$, the new CS-dependency graph does not increase the number of arcs. More sophisticated strategies like (the corresponding adaptations of) the safe transformation 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:


These 90 examples are part of the Termination Problem Data Base (TPDB, version 4.0):

http://www.lri.fr/~marche/tpdb/.

*We thank Fabian Emmes for providing this example.*
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 $(K, \mu)$. 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 $\prec$ 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 $K$ as a proof of termination of a transformed TRS $N_\Theta^\mu$ (where $\Theta$ represents the transformation). If we are able to prove termination of $N_\Theta^\mu$ (using the standard methods), then the $\mu$-termination of $K$ 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 $K$ and replacement maps $\mu$ that produce TRSs $N_\Theta^\mu$ have been investigated by Lucas (transformation L [43]), Zantema (transformation Z [63]), Ferreira and Ribeiro (transformation FR [22]), and Giesl and Middeldorp (transformations $^9$ GM, sGM, and C [30,31]), see [31,50] for recent surveys about these transformations which also include a thorough analysis about their relative power. All these transformations were considered in our experiments, so the item "Transformations" in Table 1 concentrates the joint impact of all of them.

From the benchmarks summarized in Table 1, we clearly conclude that the CSDP-framework is the most powerful technique for proving termination of CSR. Actually, all the examples that were solved by using CSRO 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 $K$ can be used to compute the list of prime numbers by using the well-known Eratosthenes sieve$^{10}$ [30]:

\[
\begin{align*}
\text{primes} & \rightarrow \text{sieve}(\text{from}(s(a(0)))) \\
\text{from}(x) & \rightarrow \text{con}(x, \text{from}(s(x))) \\
\text{head}(\text{con}(x, y)) & \rightarrow x \\
\text{sieve}(\text{con}(x, y)) & \rightarrow \text{con}(x, \text{filt}(x, \text{sieve}(y))) \\
\text{tail}(\text{con}(x, y)) & \rightarrow y \\
\text{if}(\text{true}, x, y) & \rightarrow x \\
\text{if}(\text{false}, x, y) & \rightarrow y \\
\text{filt}(s(x), \text{con}(y, z)) & \rightarrow \text{if}(\text{div}(s(x), y), \text{filt}(s(x), z), \text{con}(y, \text{filt}(s(x), z)))
\end{align*}
\]

$^9$ The labels for these transformations correspond to the ones introduced in [50].

$^{10}$ Without appropriate rules for defining symbol $\text{sieve}$, 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.
Both version of [3, 4]. We provide complete proofs for all results, 11 and also present many examples about the use of the theory.

14. Related work

proving termination of CSR

are specifically solved by them are different. Narrowing is useful for simplifying the graph, but it does not play an important speed of the CSDP technique. Furthermore, these two groups of CS processors are complementary: the extra problems that Proc an important role in our proofs. The subterm 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 Proc\(_{\text{CSNP}}\) and Proc\(_{\text{CSNP}}\) are quite efficient, but the ones that are based on simple projections for non-\(\mu\)-replacing arguments (Proc\(_{\text{CSNP}}\) and Proc\(_{\text{CSNP}}\)) 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).

Consider the replacement map \(\mu\) for the signature \(\mathcal{F}\) given by:

\[
\mu(\text{cons}) = \mu(1:) = [1] \quad \text{and} \quad \mu(f) = [1, \ldots, ar(f)] \quad \text{for all} \ f \in \mathcal{F} - \{\text{cons}, 1:\}.
\]

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 Proc\(_{\text{CSNP}}\), 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 Proc\(_{\text{CSNP}}\) and Proc\(_{\text{CSNP}}\) are quite efficient, but the ones that are based on simple projections for non-\(\mu\)-replacing arguments (Proc\(_{\text{CSNP}}\) and Proc\(_{\text{CSNP}}\)) 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, AP\(\text{ProVE}\) [32] was the only tool (besides MU-TERM) implementing specific methods for proving termination of CSR. Both AP\(\text{ProVE}\) and MU-TERM participated in the CSR subcategory of the 2007 International Termination Competition. AP\(\text{ProVE}\) participated with a termination expert for CSR which, given a CS-TRS \((\mathcal{A}, \mu)\), successively tries different transformations \(\phi\) for proving termination of CSR (which are enumerated in Remark 15, i.e., \(\phi \in \{C, FR, GM, L, sGM, Z\}\) ). It then uses a huge variety of different and complementary techniques to prove termination of rewriting (according to the DP-framework) on the obtained TRS \(\mathcal{R}_{\mu}^\phi\). Actually, AP\(\text{ProVE}\) 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 AP\(\text{ProVE}\) in the CSR category (MU-TERM was able to prove 68 of the 90 examples; AP\(\text{ProVE}\) proved 64), thus demonstrating that CSDPs are actually a very powerful technique for proving termination of CSR.

14. Related work

The first presentation of the context-sensitive dependency pairs was given in [3]. This paper is an extended and revised version of [3, 4]. We provide complete proofs for all results, 11 and also present many examples about the use of the theory. The main conceptual differences between [3,4] and this paper are the following:

1. In this paper, we have investigated two different notions of minimal non-\(\mu\)-terminating terms: the so-called strongly minimal terms \(\mathcal{T}_{\mathcal{M}, \mu}\), which are introduced in this paper) and the minimal terms \(\mathcal{M}_{\mathcal{M}, \mu}\), which were introduced in [3] and further investigated in [4]. The combined use of these notions leads to a better development of the theory. This has brought new essential results, remarkably Theorem 1, which is the basis (at the level of pure context-sensitive rewriting) of the new notions of CSDP and minimal chain.

11 We report and fix some bugs in previous papers.
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,\(^{12}\) the processors for filtering or transforming collapsing pairs (see Section 9), the use of argument filterings,\(^{13}\) the use of the subterm criterion with collapsing pairs (Theorem 13), etc.

9. Finally, for the first time, we have considered how to disprove termination of CSR within the CSDP framework (processor Proc\textsubscript{Inf} in Theorem 5).

14.1. CSDPs vs. DPs and a piece of history

The first attempt to develop a theory of dependency pairs for CSR started more than 10 years ago when the third author of this paper asked Thomas Arts (who was preparing the first presentation of the dependency pair method [9]) about the possibility of extending the dependency pair approach to CSR. Arts immediately noticed that the main problem of extending the existing results for ordinary rewriting to CSR was the possibility of having variables that are not replacing in the left-hand sides of the rules but that become replacing in the corresponding right-hand side. This is what we now call migrating variables. After this first failed attempt, the focus moved to transformations of CS-TRSs \((R, \mu)\) into ordinary TRSs \(R^\Theta\) (where \(\Theta\) represents the transformation) in such a way that termination of \(R^\Theta\) implies the \(\mu\)-termination of \(R\) [31, 50].

\(^{12}\) This is mentioned in [3, Section 4.2] but without any formal description.

\(^{13}\) This was briefly mentioned at the end of [3, Section 4.2] but was never formalized.
During the spring of 2006, mu-term was being revised in preparation for its participation in the 2006 International Termination Competition, which was organized by Claude Marché. The idea of adapting DPs to CSR came up again. A first correct version of context-sensitive dependency pairs that did not at the time consider collapsing pairs was the following:

**Definition 15** (First preliminary version of CSDPs). Let \( \mathcal{R} = (\mathcal{F}, \mathcal{R}) = (\mu \cup \mathcal{D}, \mathcal{R}) \) be a TRS and \( \mu \in \mathbb{M}_R \). Let

\[
\text{DP}_1(\mathcal{R}, \mu) = \left\{ \begin{array}{l}
\mu(f) = \mu(f) \quad \text{if } f \in \mathcal{F}, \mu^{\sharp}(f^{\sharp}) = \mu(f) \quad \text{if } f \in \mathcal{D}, \text{ and } \mu^{\sharp}(\text{MUSUBTERM}) = \emptyset,
\end{array} \right.
\]

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 \( s = f(x_1, \ldots, x_k) \) 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 \( \mathcal{R} = (\mathcal{F}, \mathcal{R}) = (\mu \cup \mathcal{D}, \mathcal{R}) \) be a TRS and \( \mu \in \mathbb{M}_R \). Let \( \text{DP}_2(\mathcal{R}, \mu) = \text{DP}_{2,\gamma}(\mathcal{R}, \mu) \cup \text{DP}_{2,\alpha}(\mathcal{R}, \mu) \) where:

\[
\begin{align*}
\text{DP}_{2,\gamma}(\mathcal{R}, \mu) &= \left\{ \begin{array}{l}
\mu^\gamma(f) = \mu(f) \quad \text{if } f \in \mathcal{F}, \mu^\gamma(f^{\sharp}) = \mu(f) \quad \text{if } f \in \mathcal{D}, \text{ and } \mu^\gamma(\text{MUSUBTERM}) = \emptyset,
\end{array} \right. \\
\text{DP}_{2,\alpha}(\mathcal{R}, \mu) &= \left\{ \begin{array}{l}
\mu^\alpha(f) = \mu(f) \quad \text{if } f \in \mathcal{F}, \mu^\alpha(f^{\sharp}) = \mu(f) \quad \text{if } f \in \mathcal{D}, \text{ and } \mu^\alpha(\text{MUSUBTERM}) = \emptyset,
\end{array} \right.
\end{align*}
\]

Here, migrating variables \( x \) are enclosed inside a term \( \text{MUSUBTERM}(x) \) which (after instantiating \( x \) by means of a substitution \( \sigma \)) would be able to connect any \( \mu \)-replacing subterm \( s = f(x_1, \ldots, x_k) \) (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 \( \text{DP}_{2,\gamma}(\mathcal{R}, \mu) \) in Definition 16 and those that were introduced to treat the migrating variables \( \text{DP}_{2,\alpha}(\mathcal{R}, \mu) \) 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:


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.
Let \( R \) be a TRS and \( \mu \in M_R \). The function symbol \( f \) hides the argument \( i \) if there is a rule \( t \rightarrow r \in R \) with \( \mu(f(t_1, \ldots, t_n)) \) or \( \mu(f) \) and \( r \) contains a defined symbol or a variable at an active position. A context \( C \) is hiding if \( C = \square \) or \( C \) has the form \( f(t_1, \ldots, t_{i-1}, C', t_{i+1}, \ldots, t_n) \) where \( f \) hides the argument \( i \) and \( C' \) is a hiding context.

The notion of CSDPs that is given in [1] is the following:

**Definition 18** [1, Definition 9]. Let \( R \) be a TRS and \( \mu \in M_R \). If \( \text{DP}_R(\emptyset, \mu) \neq \emptyset \), we introduce a fresh unhiding tuple symbol \( U \) and the following unhiding DPs:

- \( s \rightarrow U(x) \) for every \( s \rightarrow x \in \text{DP}_R(\emptyset, \mu) \),
- \( U(f(x_1, \ldots, x_n)) \rightarrow U(x_i) \) for every function symbol \( f \) of every arity \( n \) and every \( 1 \leq i \leq n \) where \( f \) hides position \( i \),
- \( U(t) \rightarrow t^2 \) for every hidden term \( t \).

Let \( \text{DP}_R(\emptyset, \mu) \) be the set of all unhiding DPs (where \( \text{DP}_R(\emptyset, \mu) = \emptyset \) whenever \( \text{DP}_R(\emptyset, \mu) = \emptyset \)). Then \( \text{DP}^\mu(R, \mu) = \text{DP}_R(\emptyset, \mu) \cup \text{DP}_R(\emptyset, \mu) \). The corresponding definition of chain is, essentially, the standard one [10], but \( \mu \)-rewriting (with \( R \)) 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 \( \mu(f(x_1^1, \ldots, x_n^1)) = \mu(f(x_1^2, \ldots, x_n^2)) \) for all \( f \in \mathcal{D} \) and \( \mu(\emptyset) = \emptyset \). A sequence of pairs \( u_1 \rightarrow v_1, u_2 \rightarrow v_2, \ldots \) from \( \mathcal{P} \) is a \( (\mathcal{P}, \mathcal{R}, \mu) \)-chain if there is a substitution \( \sigma \) with \( \sigma(x_1^1) \rightarrow v_1, \sigma(x_2^1) \rightarrow v_2, \ldots \) and \( \sigma(x) \in (\mathcal{R}, \mu) \)-nonterminating for all \( i \).

Using these definitions, a characterization of termination of CSDR is given.

**Theorem 17** [1, Theorem 12]. A TRS \( R \) is \( \mu \)-terminating if and only if there is no infinite \( (\text{DP}^\mu(R, \mu), \mathcal{R}, \mu) \)-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 15.1, 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 \( U \) instead of \( \text{HSD}_{\text{R}} \) 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 \( \mathcal{R} = \langle \mathcal{F}, \mathcal{R} \rangle \) and \( \mathcal{P} = \langle \mathcal{G}, \mathcal{P} \rangle \) be TRSs and \( \mu \in M_{\mathcal{F};\mathcal{G}} \). Let \( a \rightarrow x \in \mathcal{P} \times \mathcal{F} \) and \( \mathcal{P}_x = \{ a \rightarrow U(x) \} \)

\[ \cup \{ U(f(x_1, \ldots, x_n)) \rightarrow U(x_i) \mid f \in \mathcal{F}, 1 \leq i \leq n \text{ and } f \text{ hides } i \} \]

\[ \cup \{ U(t) \rightarrow t^2 \mid t \in \text{NT}_{\mathcal{F}} \} \]

where \( U \) is a fresh symbol. Let \( \mathcal{P}' = \langle \mathcal{G} \cup \{ U \}, \mathcal{P}' \rangle \), where \( \mathcal{P}' = \langle \mathcal{P} \rightarrow \{ a \rightarrow x \} \cup \mathcal{P}_x, \mu' \rangle \) which extends \( \mu \) by \( \mu'(U) = \emptyset \).

Then, the processor \( \text{Proc}_{\text{CS}} \) given by

\[ \text{Proc}_{\text{CS}}(\mathcal{P}, \mathcal{R}, \mu') = \{ (\mathcal{P}', \mathcal{R}, \mu') \} \]

is sound and complete.

The proof of this result would be analogous to the one for Theorem 8 with the proviso, in our definition of chain of pairs (Definition 5), that the contexts \( C \cup \{ p \} \) which are used for handling collapsing pairs are now hiding contexts. In contrast to
In this paper, we have shown that collapsing pairs are an essential part of the theoretical description of termination of CSR. Actually, Definition 18 explicitly uses them to introduce the new unhiding pairs. This shows that the most basic notion when modeling the termination behavior of CSR is that of collapsing pair and that unhiding pairs should be better considered as an ingredient for handling collapsing pairs in proofs of termination (as implemented by processor $Proc_{hCtx}$ above).

15.2. Use of CSDPs and noncollapsing CSDPs

The application of Definition 18 at the very beginning of the termination analysis of CS-TRSs (as done in [1]) often leads to obtaining a more complex dependency graph. For instance, we would replace the collapsing CSDPs (25) and (26) by the following ones:

\[
\begin{align*}
\text{T A I L}(\text{cons}(x, xx)) & \rightarrow U(x) \\
\text{T A K E}(s(n), \text{cons}(x, xx)) & \rightarrow U(x) \\
U(\text{incr}(x)) & \rightarrow U(x) \\
U(\text{incr}(\text{oddN}s)) & \rightarrow \text{INCR}(\text{oddN}s) \\
U(\text{oddN}s) & \rightarrow \text{ODDNS} \\
U(\text{rep2}(x)) & \rightarrow U(x) \\
U(\text{zip}(x, y)) & \rightarrow U(x) \\
U(\text{zip}(x, y)) & \rightarrow U(y) \\
U(\text{cons}(x, y)) & \rightarrow U(x)
\end{align*}
\]

(39) (40) (41) (42) (43) (44) (45) (46) (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 $P$ contains no collapsing pairs (as happens if Definition 18 is used to compute the dependency pairs of a CS-TRS), then Definition 19 is subsumed by our notion of chain of pairs (Definition 5). This means that, after using processor $Proc_{eColl}$ in Theorem 8 to remove collapsing pairs in the component $P$ of a CS problem $(P, R, \mu)$, 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 $P$ and $R$ in the example is not correct due to the lack of strong $\mu$-conservativeness of the $\mu$-narrowed pair in $P$. Since $P$ has no collapsing pair, one could think (following a naive 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.
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 ProcAIC 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 of 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. Soberm criterion processors.
3. Reduction pair (RP) processors with polynomial and matrix interpretations over the reals [6,7,49,51].

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 $\gamma$ in both the left- and right-hand sides, we have collapsing dependency pairs that have a single variable in the right-hand side. These variables reflect the effect of the migrating variables on the termination behavior of CSR. At the level of minimal chains, however, the contrast with the ordinary DP approach is somehow recovered by a nice symmetry arising from the central notion of hidden term (Definition 3): a noncollapsing pair $u \to v$ is followed by a pair $u' \to v'$ if $\sigma(v')$ $\mu$-rewrites into $\sigma(u')$ for some substitution $\sigma$; a collapsing pair $u \to v$ is followed by a pair $u' \to v'$ if there is a hidden term $t$ such that $\sigma(t)\gamma$ $\mu$-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 DP($\nu$, $\mu$) characterizes the $\mu$-termination of $\nu$ (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 filters 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 \( R = (F, R) \) be a TRS, \( \mu \in M_r \), and \( s \in T(F, X) \). If \( s \) is not \( \mu \)-terminating, then there is a subterm \( t \) of \( s \) such that \( t \in T_{\infty, \mu} \).

Proof. By structural induction. If \( s \) is a constant symbol, it is obvious: take \( t = s \). If \( s = f(t_1, \ldots, t_n) \), then we proceed by contradiction. If there is no subterm \( t \) of \( s \) such that \( t \in T_{\infty, \mu} \), then \( s \notin T_{\infty, \mu} \). Since \( s \) is not \( \mu \)-terminating, there is a strict \( \mu \)-replacing subterm \( t \) of \( s \) that is not \( \mu \)-terminating. By the Induction Hypothesis, there is \( t' \in T_{\infty, \mu} \) such that \( t \triangleright t' \). Then, we have \( s \triangleright t' \), thus leading to a contradiction.

Lemma 3. Let \( R = (F, R) \) be a TRS, \( \mu \in M_r \), and \( s \in T(F, X) \). If \( s \) is not \( \mu \)-terminating, then there is a \( \mu \)-replacing subterm \( t \) of \( s \) such that \( t \in M_{\infty, \mu} \).

Proof. By structural induction. If \( s \) is a constant symbol, it is obvious: take \( t = s \). If \( s = f(t_1, \ldots, t_n) \), then we proceed by contradiction. If there is no \( \mu \)-replacing subterm \( t \) of \( s \) such that \( t \in M_{\infty, \mu} \), then there is a strict \( \mu \)-replacing subterm \( t \) of \( s \) which is not \( \mu \)-terminating. We have \( t = s_p \) for some \( p \in \text{Var}(s) - \{ \lambda \} \). By the Induction Hypothesis, \( t \) contains a \( \mu \)-replacing subterm \( t' \) which belongs to \( M_{\infty, \mu} \), i.e., \( t' = s_{p} \) for some \( q \in \text{Var}(s) \). By Proposition 1, \( p \cdot q \in \text{Var}(s) \). Thus, \( t' \) is a \( \mu \)-replacing subterm of \( s \) that belongs to \( M_{\infty, \mu} \), thus leading to a contradiction.

Lemma 4. Let \( R \) be a TRS, \( \mu \in M_r \), and \( t \in M_{\infty, \mu} \). If \( t \triangleright_{\infty, \mu} u \) and \( u \) is non-\( \mu \)-terminating, then \( u \in M_{\infty, \mu} \).

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 \( u \) are \( \mu \)-terminating. Since \( u \) is non-\( \mu \)-terminating, \( u \in M_{\infty, \mu} \).

16.1. Proofs of Section 3.2

Lemma 5. Let \( R = (F, R) \) be a TRS and \( \mu \in M_r \). Let \( t \in T(F, X) \) and \( \sigma \) be a substitution. If there is a rule \( l \rightarrow r \in R \) such that \( \sigma(l) \triangleright t \) and \( \sigma(r) \triangleright t \), then there is no \( x \in \text{Var}(r) \) such that \( \sigma(x) \triangleright t \). Furthermore, there is a term \( t' \in H \{ t \} \) 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 \( \sigma(x) \triangleright t \), we have that \( \sigma(r) \triangleright_{\mu} t \), then there is a nonvariable and non-\( \mu \)-replacing position \( p \in \text{Var}(r) - \{ \text{Var}(l) \} \) of \( r \) such that \( \sigma(r)p = t \). Then, we let \( t' = r_{p} \). Note that \( t' \in H \{ t \} \).

Lemma 6. Let \( R \) be a TRS and \( \mu \in M_r \). Let \( A \) be a \( \mu \)-rewrite sequence \( t_1 \Rightarrow t_2 \Rightarrow \cdots \Rightarrow t_o \) with \( t_i \in M_{\infty, \mu} \) for all \( i, 1 \leq i \leq o \). If there is a term \( t \in M_{\infty, \mu} \) such that \( t_1 \triangleright_{\mu} t \) and \( t_o \triangleright_{\mu} t \), then \( t = \sigma(s) \) for some \( s \in D H \{ t \} \) and substitution \( \sigma \).

Proof. By induction on \( o \):

1. If \( o = 1 \), then it is vacuously true because \( t_1 \triangleright_{\mu} t \) and \( t_o \triangleright_{\mu} t \) do not simultaneously hold.
2. If \( n = 1 \), then we assume that \( t_1 \not\triangleright \gamma \) and \( t_0 \triangleright \gamma \). We consider two cases:

(a) If \( t_{n-1} \triangleright \gamma \), then by the induction hypothesis the conclusion follows.

(b) If \( t_{n-1} \not\triangleright \gamma \) does not hold, then, since \( t_{n-1} \in M_{\lambda_0, \mu} \) in the hypothesis implies that \( t \not\in M_{\lambda_0, \mu} \), we have that \( t_{n-1} \not\triangleright \gamma \). Let \( t \rightarrow t' \in R \) be such that \( t_{n-1} = C(\sigma(t)) \) and \( t_0 = C(\sigma(t)) \) for some context \( C(t) \) and substitution \( \sigma \). Then, in particular, \( \sigma(t) \not\triangleright \gamma \) and, since \( t_0 \triangleright \gamma \), there must be \( \sigma(r) \not\triangleright \gamma \). Thus, by Lemma 5 we conclude that \( t = \sigma(s) \) for some \( s \in D \) and substitution \( \sigma \). Since \( t \in M_{\lambda_0, \mu} \), it follows that \( \rho(t) = \rho(s) \in D \). Thus, \( s \in D \). □

**Proposition 4.** Let \( R \) be a TRS and \( \mu \in M_\lambda \). Consider a finite or infinite sequence of the form \( t_1 \xrightarrow{\rho} s_1 \xrightarrow{\rho} t_2 \xrightarrow{\rho} \cdots \) with \( t_i, \mu \in M_{\lambda_0, \mu} \) for all \( i \geq 1 \). If there is a term \( t \in M_{\lambda_0, \mu} \) such that \( t \not\triangleright \gamma \) for some \( i \geq 1 \), then \( t_1 \not\triangleright \gamma \) or \( t = \sigma(s) \) for some \( s \in D \) and substitution \( \sigma \).

**Proof.** By induction on \( i \):

1. If \( i = 1 \), it is trivial.

2. If \( i > 1 \) and \( t_i \triangleright \gamma \), then we consider two cases:

(a) If \( t_{i-1} \triangleright \gamma \), then by the induction hypothesis, the conclusion follows.

(b) If \( t_{i-1} \not\triangleright \gamma \) does not hold, then let \( t \rightarrow t' \in R \) and \( \sigma \) be such that \( t_{i-1} = \sigma(t) \) and \( t_{i-1} = \sigma(t) \not\triangleright \gamma \). Since \( t_{i-1} \not\triangleright \gamma \), \( t_0 \not\triangleright \gamma \). We consider two cases:

(A) If \( \sigma(t) \triangleright \gamma \), then, since \( \sigma(t) \in M_{\lambda_0, \mu} \), the case \( \sigma(t) \not\triangleright \gamma \) is excluded and the only possibility is that \( t \not\triangleright \gamma \). Then, since \( \sigma(t) = t_0 \triangleright \gamma \), \( \sigma(t) = t_0 \not\triangleright \gamma \) and \( \sigma(t) \not\triangleright \gamma \). By Lemma 5 we conclude that \( t = \sigma(s) \) for some \( s \in D \) and substitution \( \sigma \).

(B) If \( \sigma(t) \not\triangleright \gamma \), then, by applying Lemma 4 and Lemma 6 to the \( \mu \)-rewrite sequence \( t_1 \xrightarrow{\rho} t_2 \xrightarrow{\rho} \cdots t_n \), the conclusion follows. □

**References**


8.12 Proving Termination in the Context-Sensitive Dependency Pair Framework

Proving Termination in the Context-Sensitive Dependency Pair Framework

Raul Gutiérrez1 and Salvador Lucas1
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 \( \mu(f) \subseteq \{1, \ldots, \text{arity}(f)\} \) for every function symbol \( f \) of arity \( \text{arity}(f) \) in the signature \( \mathcal{F} \) is used to discriminate the argument positions on which the rewriting steps are allowed. In this way, a terminating behavior of (context-sensitive) computations with Term Rewriting Systems (TRSs) can be obtained. CSR has shown useful to model evaluation strategies in programming languages. In particular, it is an essential ingredient to analyze the termination behavior of programs in programming languages (like CafeOBJ, Maude, OBJ, etc.) which implement recent presentations of rewriting logic like the Generalized Rewrite Theories [2], see [3–5].

* Partially supported by the EU (FEDER) and the Spanish MEC/MICINN, under grant TIN 2007-68093-C02-02.
Example 1. Consider the following TRS in [6]:

\[
\begin{align*}
gt(0, y) & \rightarrow \text{false} \quad p(0) \rightarrow 0 \\
gt(s(x), 0) & \rightarrow \text{true} \quad p(s(x)) \rightarrow x \\
gt(s(x), s(y)) & \rightarrow gt(x, y) \\
\text{if}(\text{true}, x, y) & \rightarrow x \\
\text{div}(0, s(y)) & \rightarrow 0 \\
\text{div}(s(x), s(y)) & \rightarrow s(\text{div}(\text{minus}(x, y), s(y))) \\
\text{minus}(0, 0) & \rightarrow \text{true} \\
\text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0) & \rightarrow \text{false} \\
\end{align*}
\]

with \(\mu(\text{if}) = \{1\}\) and \(\mu(f) = \{1, \ldots, \text{arity}(f)\}\) 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:

\[
\text{minus}(0, 0) \rightarrow \text{true} \\
\text{if}(\text{gt}(0, 0), \text{minus}(0, 0), 0) \rightarrow \text{false} \\
\text{if}(\text{true}, x, y) \rightarrow x \\
\text{if}(\text{false}, x, y) \rightarrow y \\
\text{minus}(0, 0) \rightarrow \text{false} \\
\text{if}(\text{false}, x, y) \rightarrow y \\
\text{div}(s(x), s(y)) \rightarrow s(\text{div}(\text{minus}(x, y), s(y))) \\
\]

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 \(R\) 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\(^{1}\) \(g\). 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 \(u = \ell(t_1, \ldots, t_n)\) if \(\ell = f(t_1, \ldots, t_n)\) and \(v = g^1(s_1, \ldots, s_m)\) if \(s = g(s_1, \ldots, s_m)\). Note that, if \(g\) is a defined symbol and \(\ell\) is marked. In practice, we often capitalize \(f\) and use \(F\) instead of \(\ell\) in our examples. For this reason, the dependency pair technique starts by considering a new TRS \(\text{DP}(R)\) which contains all these dependency pairs for each \(\ell \rightarrow r \in R\). The rules in \(\text{DP}(R)\) determine the so-called dependency chains whose finiteness characterizes termination of \(R\) [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 \(u \rightarrow v\) 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., \(u = \ell\) but \(x\) is a migrating variable which is \(\mu\)-replacing in \(r\) but which only occurs at \(\mu\)-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

\(^{1}\) A symbol \(g \in F\) is defined in \(R\) if there is a rule \(\ell \rightarrow r \in R\) whose left-hand side \(\ell\) is of the form \(g(t_1, \ldots, t_k)\) for some \(k \geq 0\).
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 loose the intuition of what collapsing pairs mean in a context-sensitive rewriting chain. And understanding the new dependency graph is harder.

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

\begin{align*}
&\text{GT}(s(x), x(y)) \rightarrow \text{GT}(x, y) \quad (1) \\
&\text{M}(x, y) \rightarrow \text{GT}(y, 0) \quad (2) \\
&\text{D}(s(x), x(y)) \rightarrow \text{M}(x, y) \quad (3) \\
&\text{IF}(\text{true}, x, y) \rightarrow U(x) \quad (4) \\
&\text{IF}(\text{false}, x, y) \rightarrow U(y) \quad (5) \\
&U(p(x)) \rightarrow P(x) \quad (6)
\end{align*}

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{align*}
&\text{GT}(s(x), x(y)) \rightarrow \text{GT}(x, y) \quad (1) \\
&\text{M}(x, y) \rightarrow \text{IF}(\text{gt}(y, 0), \text{minus}(p(x), p(y)), x) \quad (7) \\
&\text{D}(s(x), x(y)) \rightarrow \text{M}(x, y) \quad (3) \\
&\text{IF}(\text{true}, x, y) \rightarrow x \quad (14) \\
&\text{IF}(\text{false}, x, y) \rightarrow y \quad (15)
\end{align*}

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

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

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

2 Preliminaries

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

In the following, we introduce some notions and notation about CSR [1]. A mapping $\mu : F \rightarrow \varphi(\mathbb{N})$ is a replacement map if $\forall f \in F, \mu(f) \subseteq \{1, \ldots, \varphi(f)\}$. Let $M_R$ be the set of all replacement maps (or $M_R$ for the replacement maps of a TRS $R = (F, R)$). The set of μ-replacing positions $\Pos_{\mu}(t)$ of $t \in T(F, X)$ is: $\Pos_{\mu}(t) = \{\Lambda\}$, if $t \in X$ and $\Pos_{\mu}(t) = \{\Lambda\} \cup \bigcup_{p \in \Pos(t)} \mu(p).\Pos_{\mu}(t)_p$, if $t \notin X$. The set of μ-replacing variables of $t$ is $\Var_{\mu}(t) = \{x \in \Var(t) \mid \exists p \in \Pos_{\mu}(t), t|_p = x\}$ and $\Var_{\mu}(t) = \{x \in \Var(t) \mid \exists p \in \Pos_{\mu}(t), t|_p = x\} \setminus \Var_{\mu}(t)_p$. Note that $\Var_{\mu}(t)$ and $\Var_{\mu}(t)$ do not need to be disjoint. The μ-replacing subterm relation $\supseteq_{\mu}$ is given by $t \supseteq_{\mu} s$ if there is $p \in \Pos_{\mu}(t)$ such that $s = t|_p$. We write $t \supset_{\mu} s$ if $t \supseteq_{\mu} s$ and $t \neq s$. We write
Proving Termination in the CSDP Framework

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., $t \not\rightarrow_{\mu} s$) 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 $T_{\infty, \mu}$ be a set of strongly minimal non-$\mu$-terminating terms in the following sense: $t$ belongs to $T_{\infty, \mu}$ if $t$ is non-$\mu$-terminating and every strict $\mu$-replacing subterm $u$ (i.e., $t \not\rightarrow u$) is $\mu$-terminating. It is obvious that $\text{root}(t) \in D$ for all $t \in T_{\infty, \mu}$.

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 $t_0 \in T_{\infty, \mu}$ of $t$ which starts a minimal infinite $\mu$-rewrite sequence of the form $t_0 \not\rightarrow_{\mu} \cdots \not\rightarrow_{\mu} \sigma_1(t_1) \not\rightarrow_{\mu} \sigma_1(t_1) \not\rightarrow_{\mu} t_1 \not\rightarrow_{\mu} \sigma_2(t_2) \not\rightarrow_{\mu} \cdots$ where $t_0, \sigma_i(t_i) \in \mathcal{M}_{\infty, \mu}$ 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 $\ell_i \not\rightarrow r_i$.
- 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 $\ell_i \not\rightarrow r_i$, i.e., $x_i \in \text{Var}^M(r_i) \setminus \text{Var}^M(\ell_i)$. 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]: $\text{Ren}^\mu(t)$, which independently renames all occurrences of $\mu$-replacing variables by using new fresh variables which are not
in \( \text{Var}(t) \), and \( \text{Narr}_\mu^2(t) \), which indicates whether \( t \) is \( \mu \)-narrowable\(^2\) \((\text{w.r.t. the intended TRS } \mathcal{R})\).

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

\[
\mathcal{N}^i(\mathcal{R}, \mu) = \{ t \in \mathcal{H}(\mathcal{R}, \mu) \mid \text{root}(t) \in \mathcal{D} \text{ and } \text{Narr}_\mu^2(\text{Ren}^i(t)) \}
\]

**Definition 2 (Hiding Context).** Let \( \mathcal{R} \) be a TRS and \( \mu \in M_\mathfrak{R} \). A function symbol \( f \) hides position \( i \) in the rule \( \ell \to r \in \mathcal{R} \) if \( f(r_1, \ldots, r_n) \) for some terms \( r_1, \ldots, r_n \), and there is \( i \in \mu(f) \) such that \( r_i \) contains a \( \mu \)-replacing defined symbol \( i.e., \text{Pos}_\mu^2(r_i) \neq \emptyset \) or a variable \( x \in (\text{Var}_\mathfrak{F}(\ell) \cap \text{Var}_\mathfrak{F}(r)) \setminus (\text{Var}^s(\ell) \cup \text{Var}^s(r)) \) which is \( \mu \)-placing in \( r_i \) \((i.e., x \in \text{Var}^s(r_i))\). A context \( C[\Box] \) is hiding \([6]\) if \( C[\Box] = \Box \), or \( C[\Box] = f(t_1, \ldots, t_{n+1}, C'[\Box], t_{n+2}, \ldots, t_k) \), where \( f \) hides position \( i \) and \( C'[\Box] \) is a hiding context.

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

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

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

\[
t = t_0 \underbrace{\geq_{\mathcal{R}, \mu} \sigma_1(t_1) \geq_{\mathcal{R}, \mu} \sigma_2(t_2) \geq_{\mathcal{R}, \mu} \cdots}_{\text{where, for all } i \geq 1, t_i \to r_i \in \mathcal{R} \text{ are rewrite rules, } \sigma_i \text{ are substitutions, and terms } t_i \in \mathcal{M}_{\omega, \mu} \text{ are minimal non-} \mu \text{-terminating terms such that either}}
\]

1. \( t_i = \sigma_i(s_i) \) for some nonvariable term \( s_i \) such that \( r_i \geq_{\mathcal{R}, \mu} s_i \), or
2. \( \sigma_i(s_i) = \theta_i(C_i[t'_i]) \) and \( t_i = \theta_i(t'_i) \) for some variable \( s_i \in \text{Var}^s(r_i) \setminus \text{Var}^s(t_i) \), \( \theta'_i \in \mathcal{N}^i(\mathcal{R}, \mu) \), hiding context \( C_i[\Box] \), 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.

\(^2\)A term \( s \)-\( \mu \)-narrow to the term \( t \) if there is a nonvariable position \( p \in \text{Pos}_\mu^2(s) \) and a rule \( \ell \to r \) such that \( s_p \) and \( \ell \) unify with \( \text{mgu } \sigma \), and \( t = \sigma(s[p]) \).
Definition 3 (Context-Sensitive Dependency Pairs [9]). Let $R = (F, R)$ be a TRS and $\mu \in M_F$. We define $DP(R, \mu) = DP_F(R, \mu) \cup DP_A(R, \mu)$ to be set of context-sensitive dependency pairs (CSDPs) where:

$$DP_F(R, \mu) = \{ \ell \rightarrow s^i | \ell \rightarrow r \in R, r \notin s, \text{root}(s) \in D, \ell \notin s, \text{Narb}_F(Ren^\ast(s)) \}$$

$$DP_A(R, \mu) = \{ \ell^i \rightarrow x | \ell \rightarrow r \in R, r \in \text{Var}^\ast(r) \setminus \text{Var}^\ast(\ell) \}$$

We extend $\mu \in M_F$ into $\mu^+ \in M_{F,\mu}$ by $\mu^+(f) = \mu(f)$ if $f \in F$ and $\mu^+(f) = \mu(f)$ if $f \notin D$.

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 $R$ be a TRS and $\mu \in M_R$. We define $\text{unh}(R, \mu)$ as the TRS consisting of the following rules:

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

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

\[
\begin{align*}
\text{minus}(p(x), p(y)) & \rightarrow M(p(x), p(y)) \quad (16) \\
p(x) & \rightarrow P(x) \quad (17) \\
\text{minus}(x, y) & \rightarrow y \quad (18) \\
\text{minus}(x, y) & \rightarrow x \quad (19)
\end{align*}
\]

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 $S$, we let $S_{>\mu}$ be the rules from $S$ of the form $s \rightarrow t\in S$ and $s \ni_{\mu} t$; and $S_\mu = S \setminus S_{>\mu}$.

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

1. if $v_i \notin \text{Var}(u_i) \setminus \text{Var}^\ast(u_i)$, then $\sigma(v_i) = t_i \rightarrow_{\mu} \sigma(u_{i+1})$, and
2. if $v_i \in \text{Var}(u_i) \setminus \text{Var}^\ast(u_i)$, then $\sigma(v_i) \rightarrow_{\mu} t_i \rightarrow_{\mu} \sigma(u_{i+1})$.

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

Notice that if rules $f(x_1, \ldots, x_k) \rightarrow x_i$ for all $f \in D$ and $i \in \mu(f)$ (where $x_1, \ldots, x_k$ are variables) are used in Item 1 of Definition 4, then Definition 5 yields the notion of chain in [9]; and if, additionally, rules $f(x_1, \ldots, x_k) \rightarrow f(x_1, \ldots, x_k)$ for all $f \in D$ 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 $R$ be a TRS and $\mu \in M_R$. A CS-TRS $(R, \mu)$ is terminating if and only if there is no infinite $(DP(R, \mu), R, \text{unh}(R, \mu), \mu^+)$-chain.
5 Context-Sensitive Dependency Pair Framework

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

**Definition 6 (CS Problem and CS Processor).** A CS problem \( \tau \) is a tuple \( \tau = (P, R, S, \mu) \), where \( R, P \) and \( S \) are TRSs, and \( \mu \in M_{\emptyset, \emptyset, \emptyset} \). The CS problem \( (P, R, S, \mu) \) is finite if there is no infinite \((P, R, S, \mu)\)-chain. The CS problem \( (P, R, S, \mu) \) is infinite if \( R \) is non-\( \mu \)-terminating or there is an infinite minimal \((P, R, S, \mu)\)-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 \( \text{Proc}(\tau) \neq \emptyset \) 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 \( \text{Proc}(\tau) = \emptyset \) or \( \exists \tau' \in \text{Proc}(\tau) \) such that \( \tau' \) is infinite.

In order to prove the \( \mu \)-termination of a TRS \( R \), we adapt the result from \cite{12} to CSR.

**Theorem 3 (CSDP Framework).** Let \( R \) be a TRS and \( \mu \in M_\emptyset \). We construct a tree whose nodes are labeled with CS problems or “yes” or “no”, and whose root is labeled with \((\text{DP}(R, \mu), R, \text{unhr}(R, \mu), \mu^\text{r})\). For every inner node labeled with \( \tau \), there is a sound processor \( \text{Proc} \) satisfying one of the following conditions:

1. \( \text{Proc}(\tau) = \emptyset \) 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. \( \text{Proc}(\tau) \neq \emptyset \), \( \text{Proc}(\tau) \neq \emptyset \), 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 \( R \) is \( \mu \)-terminating. Otherwise, if there is a leaf labeled with “no” and \( \text{Proc}(\tau) \neq \emptyset \), then \( R \) is not \( \mu \)-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 \cite{6} into our framework. The pairs in a CS-TRS \( (P, \mu) \), where \( P = (G, P) \), are partitioned as follows:

\( P_X = \{ u \rightarrow v \in P \mid v \in \text{Var}(u) \cup \text{Var}(u) \} \) and \( P_U = P \setminus P_X \).

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

- \( u \rightarrow U(x) \) for every \( u \rightarrow x \in P_X \),
- \( U(s) \rightarrow U(t) \) for every \( s \rightarrow t \in S_{PS} \), and
Proving Termination in the CSDP Framework

- U(s) → t for every s → t ∈ S♯.

Here, U is a new fresh symbol. Let P′ = (G ∪ {U}, P′) where P′ = (P ∖ P♯) ∪ P0, and µ′ extends µ by µ′(U) = ∅. The processor ProcColl given by ProcColl(τ) = {⟨P′, R, ∅, µ′⟩} 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 P♯ containing the rules whose migrating variables occur on non-µ-replacing immediate subterms in the left-hand side is P♯1 = {f(u1, ..., uk) → x ∈ P♯ | ∃i ≤ k, i ̸∈ m(t), x ∈ Var(u_i)}.

Theorem 5 (Basic CS Processor for Collapsing Pairs). Let τ = (P, R, S, µ) be a CS problem where R = (C ⊕ D, R) and S = (H, S). Assume that (1) all the rules in S1 are noncollapsing, i.e., for all s → t ∈ S1, t ̸∈ X (2) {root(t) | s → t ∈ S1} and (3) all s → t ∈ S♯, we have that s = {s1, ..., sk} and t = g(s1, ..., sk) for some k ∈ N, function symbols f, g ∈ H, and terms s1, ..., sk. Then, the processors ProcColl given by

ProcColl(τ) = \[ \begin{cases} \emptyset & \text{if } P = P_{k1} \\ \{(P, R, S, \mu)\} & \text{otherwise} \end{cases} \]

is sound and complete.

Example 5. (Continuing Example 1) Consider the CS problem τ = (P, R, S, µ) where P♯ = {14, 15} and S♯ = {16, 18, 19}. We can apply ProcColl(τ) to conclude that the CS problem τ is finite.

5.2 Context-Sensitive Dependency Graph

In the DP-approach [8, 12], a dependency graph is associated to the TRS R. The nodes of the graph are the dependency pairs in DP(R) and there is an arc from a dependency pair u → v to a dependency pair u′ → v′ if there are substitutions θ and θ′ such that θ(v) → R θ′(v′). In our setting, we have the following.

Definition 7 (Context-Sensitive Graph of Pairs). Let (P, R, S) be TRSs and µ ∈ M(R, P, S). The context-sensitive (CS) graph G(P, R, S, µ) has P as the set of nodes. Given u → v, u′ → v′ ∈ P, there is an arc from u → v to u′ → v′ if there is a µ-term (P, R, S, µ)-chain for some substitution σ for the use of SCCs for dealing with CS problems.
Theorem 6 (SCC Processor). Let \( \tau = (P, R, S, \mu) \) be a CS problem. Then, the processor \( \text{Proc}_{\text{SCC}} \) given by

\[
\text{Proc}_{\text{SCC}}(\tau) = \{(Q, R, S_Q, \mu) : Q \text{ contains the pairs of an SCC in } G(P, R, S, \mu) \}
\]

where \( S_Q \) are the rules from \( S \) involving a possible \((Q, R, S, \mu)\)-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 \( \text{TCap}_G^\mu \) given by

\[
\text{TCap}_G^\mu = \left\{ y^i = s^i \text{ if } i \not\in \mu(f) \text{ and } [s^i] = s \text{ if } i \in \mu(f). \right. \\
\text{otherwise}
\]

where \( y \) is a new fresh variable, \( [s^i] = \text{TCap}_G^\mu(s) \) if \( i \in \mu(f) \) and \( [s^i] = s \) if \( i \not\in \mu(f) \). We assume that \( f \) shares no variable with \( G \) when the unification is attempted.

Definition 9 (Estimated CS Graph of Pairs). Let \( \tau = (P, R, S, \mu) \) be a CS problem. The estimated CS graph \( \text{EG}(P, R, S, \mu) \) is the graph \( G(P, R, S, \mu) \) restricted to the nodes and arcs which connect them as follows:

1. there is an arc from \( u \rightarrow v \in P \) to \( u' \rightarrow v' \in P \) if \( \text{TCap}_G^\mu(v) \) and \( u' \) unify, and
2. there is an arc from \( u \rightarrow v \in P \) to \( u' \rightarrow v' \in P \) if there is a \( s \rightarrow t \in S_\mu \) such that \( \text{TCap}_G^\mu(t) \) and \( u' \) unify.

We have the following.

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

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

Theorem 8 (SCC Processor using \( \text{TCap}_G^\mu \)). Let \( \tau = (P, R, S, \mu) \) be a CS problem. The CS processor \( \text{Proc}_{\text{SCC}} \) given by

\[
\text{Proc}_{\text{SCC}}(\tau) = \{(Q, R, S_Q, \mu) : Q \text{ contains the pairs of an SCC in } \text{EG}(P, R, S, \mu) \}
\]

where

\[
S_Q = \emptyset \text{ if } \mathcal{Q}_S = \emptyset.
\]
In Figure 1 (right) we show Example 6. The graph has three SCCs $P_1 = \{(1)\}$, $P_2 = \{(8)\}$, and $P_3 = \{(7), (14), (15)\}$. If we apply the CS processor $Proc_{SCC}$ to the initial CS problem $(DP(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^3)$ for $\mathcal{R}$ in Example 1, then we obtain the problems: $(P_1, \mathcal{R}, \varnothing, \mu^3)$, $(P_2, \mathcal{R}, \varnothing, \mu^3)$, $(P_3, \mathcal{R}, \mathcal{S}_1, \mu^3)$, where $\mathcal{S}_1 = \{(16), (18), (19)\}$.

### 5.3 Reduction Triple Processor

A $\mu$-reduction pair $(\succeq, \supseteq)$ consists of a stable and $\mu$-monotonic\(^3\) quasi-ordering $\succeq$ and a well-founded stable relation $\supseteq$ on terms in $T(F, \Lambda)$ which are compatible, i.e., $\succeq \circ \supseteq \supseteq \circ \varnothing$ or $\varnothing \supseteq \supseteq \supseteq$ [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 $\succeq$ of reduction pairs $(\succeq, \supseteq)$ which are used in [7,9] are required to have the $\mu$-subterm property, i.e., $\succeq \circ \varnothing \supseteq \varnothing$. This is equivalent to impose $f(x_1, \ldots, x_i) \succeq x_i$ for all projection rules $f(x_1, \ldots, x_i) \rightarrow x_i$, with $f \in F$ and $i \in \mu(f)$. This is similar for markings: in [7] we have to ensure that $f(x_1, \ldots, x_i) \succeq f(x_1, \ldots, x_i)$ for all defined symbols $f$ in the signature. In [9], thanks to the notion of hidden term, we relaxed the last condition: we require $t \succeq t^f$ for all (narrowable) hidden terms $t$. In [6], thanks to the notion of hiding context, we only require that $\succeq$ is compatible with the projections $f(x_1, \ldots, x_i) \rightarrow x_i$ for those symbols $f$ and positions $i$ such that $f$ hides position $i$. However, this information is implicitly encoded as (new) pairs $U((f(x_1, \ldots, x_i)) \rightarrow U(x_i)$ in the set $P$. The strict component $\supseteq$ of the reduction pair $(\succeq, \supseteq)$ is used with these new pairs now.

In this paper, since the rules in $S$ 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 $S$ 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 $(\succeq, \supseteq, \preceq)$ now, where $(\succeq, \supseteq)$ is a $\mu$-reduction pair and $\preceq$ is a stable relation on terms which is compatible with $\succeq$ or $\supseteq$, i.e., $\succeq \circ \varnothing \subseteq \varnothing$ or $\varnothing \circ \supseteq \subseteq \supseteq$.

**Theorem 9 ($\mu$-Reduction Triple Processor).** Let $\tau = (P, R, S, \mu)$ be a CS problem. Let $(\succeq, \supseteq, \preceq)$ be a $\mu$-reduction triple such that

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


1. \( P \subseteq \geq \cup \bowtie, R \subseteq \geq, \) and
2. whenever \( P_X \neq \emptyset \) we have that \( S \subseteq \geq \cup \bowtie \cup \geq. \)

Let \( P \bowtie = \{ u \to v \in P \mid u \bowtie v \} \) and \( S_2 = \{ s \to t \in S \mid s \bowtie t \}. \) Then, the processor \( \text{Proc}_{RT} \) given by

\[
\text{Proc}_{RT}(\tau) = \begin{cases} 
\{(P \setminus P_\bowtie, R, S \setminus S_\bowtie, \mu) \} & \text{if (1) and (2) hold} \\
\emptyset & \text{otherwise}
\end{cases}
\]

is sound and complete.

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

Example 7. (Continuing Example 6) Consider the CS problem \( \tau = (P_3, R, S_3, \mu^3) \) where \( P_3 = \{(7), (14), (15)\}, S_3 = \{(16), (18), (19)\} \) and \( R \) is the TRS in Example 1. If we apply \( \text{Proc}_{RT} \) to the CS problem \( \tau \) by using the \( \mu \)-reduction triple (\( \geq, >, \bowtie \)) where \( \geq \) and \( > \) are the orderings induced by the following polynomial interpretation (see [17] for missing notation and definitions):

\[
\begin{align*}
\{f(x, y, z)\} &= (1/2 \times x) + y + z & \{\text{minus}(x, y)\} &= (2 \times x) + (2 \times y) + 1/2 \\
\{g(x)\} &= (1/2 \times x) & \{0\} &= 0 \\
\{\text{false}\} &= 0 & \{\text{true}\} &= (2 \times x) + 2 \\
\{M(x, y)\} &= (2 \times x) + (1/2 \times y) & \{\text{IF}(x, y, z)\} &= (1/2 \times x) + y + z
\end{align*}
\]

then, we have \( [\ell] \geq [r] \) for all (usable) rules in \( R \) and, for the rules in \( P_3 \) and \( S_3, \) we have

\[
\begin{align*}
[M(x, y)] &\geq [f(g(y, 0), \text{minus}(p(x), p(y)), z)] & [\text{minus}(p(x), p(y))] &\geq [M(p(x), p(y))] \\
[f(\text{false}, x, y)] &\geq [x] & [\text{minus}(r, 0)] &\geq [r]
\end{align*}
\]

Then, we get \( \text{Proc}_{RT}(\tau) = \{((7), (15), R, \{(16)\}, \mu^3)\}. \)

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 \( S. \) Before introducing it, we need the following definition.

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

\[
\text{Root}(R) = \{\text{root}(\ell) \mid \ell \to r \in R\} \cup \{\text{root}(r) \mid \ell \to r \in R, r \notin X\}
\]
Definition 11 (Simple Projection). Let \( R \) be a TRS. A simple projection for \( R \) is a mapping \( \pi \) that assigns to every \( k \)-ary symbol \( f \in \text{Root}(R) \) an argument position \( i \in \{1, \ldots, k\} \). This mapping is extended to terms by

\[
\pi(t) = \begin{cases} 
\pi(t[i]) & \text{if } t = f(t_1, \ldots, t_k) \text{ and } f \in \text{Root}(R) \\
    t & \text{otherwise}
\end{cases}
\]

Theorem 10 (Subterm Processor). Let \( \tau = (P, R, S, \mu) \) be a CS problem where \( R = (F, R) = (\mathcal{G} \cup D, R) \), \( P = (\mathcal{G}, P) \) and \( S = (\text{H}, S) \). Assume that \( (1) \) Root\((P) \cap D = \emptyset \), \( (2) \) the rules in \( P_D \cup S_G \) are noncollapsing, \( (3) \) for all \( s_i \rightarrow t_i \in S_{D_D} \), root\((s_i)\), root\((t_i)\) \( \notin \text{Root}(P) \) and \( (4) \) for all \( s_i \rightarrow t_i \in S_{D_D} \), root\((s_i)\) \( \notin \text{Root}(P) \) and root\((t_i)\) \( \notin \text{Root}(P) \). Let \( \pi \) be a simple projection for \( P \). Let \( P_{\mu,D} = \{ u \rightarrow v \in P \mid \pi(u) \supseteq \mu(v) \} \) and \( S_{\mu,D} = \{ s \rightarrow t \in S \mid \pi(s) \supseteq \mu(t) \} \). Then, \( \text{Proc}_{\text{subterm}} \) given by

\[
\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} 
\{ \{ P \setminus P_{\mu,D} \}, R, S \setminus S_{\mu,D} \} & \text{if } \pi(P) \subseteq \mu \\
\text{otherwise}
\end{cases}
\]

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 \( \langle P_5, R, S_5, \mu \rangle \) where \( P_5 = \{(7), (15)\} \) and \( S_5 = \{(16)\} \). We can apply the subterm processor \( \text{Proc}_{\text{subterm}} \) by using the projection \( \pi(F) = 4 \) and \( \pi(M) = 1 \):

- \( \pi(M(x), y) = x \supseteq \mu \) \( x = \pi(x) \)
- \( \pi(\text{if}(x, y, 0), \text{if}(x, y, 0)) = \pi(x) \supseteq \mu \) \( \pi(x) = \pi(x) \)
- \( \pi(\text{if}(\text{false}(x, y), 0), \text{if}(\text{false}(x, y), 0)) = \pi(x) \supseteq \mu \) \( \pi(x) = \pi(x) \)

We obtain the CS problem \( \tau' = \{(7), (15)\}, R, \emptyset, \mu \) 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 [5].

```maude
fmod LengthOfFiniteLists is
  sorts Nat, NatList, .
  subsort NatList < NatList, .
  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(0, L)) = s(length(L)) .
endfs

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 Base4 (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 paper5.

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 systems6 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

---

1 http://www.lri.fr/~marche/tpdb/
2 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/nuterm/benchmarks/benchmarks-csr/benchmarks.html
3 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)

<table>
<thead>
<tr>
<th>Tool</th>
<th>Version</th>
<th>Proved</th>
<th>Average time</th>
</tr>
</thead>
<tbody>
<tr>
<td>AProVE</td>
<td>34/37</td>
<td>3.084s</td>
<td></td>
</tr>
<tr>
<td>Jambox</td>
<td>28/37</td>
<td>2.292s</td>
<td></td>
</tr>
<tr>
<td>mu-term</td>
<td>34/37</td>
<td>1.277s</td>
<td></td>
</tr>
<tr>
<td>VMTL</td>
<td>29/37</td>
<td>6.708s</td>
<td></td>
</tr>
</tbody>
</table>

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

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

8 Related Work

In [6], a transformation of collapsing pairs into ‘ordinary’ (i.e., noncollapsing) pairs is introduced by using the new notion of ‘hiding context’ [6, Definition 7]. We easily and naturally included such a transformation as a new processor ProcColl 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 ProcColl above). Furthermore, the application of such a transformation in the very beginning of the termination analysis of CS-TRSs (as done in [6]) typically leads to obtain a more complex dependency graph (see in Figure 1 (left)) which, as witnessed by our experimental

\(^7\) A complete report of our experiments can be found in http://zenon.dsic.upv.es/muterm/benchmarks/
analysis in Section 7, can be more difficult to analyze when proving termination in practice.

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

9 Conclusions

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

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

References


8.13 Proving Termination Properties with MU-TERM

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 sub-types, 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 sort- ing information is taken into account but which is nonterminating as a TRS (i.e., disregarding sort information) [18]. The predicate \texttt{is-even} tests whether an integer number is even. When disregarding any information about sorts, the program \texttt{EVEN} is not terminating due to the last rule for \texttt{is-even}, which specifies a recursive call to \texttt{is-even}. However, when sorts are considered and the hierarchy among them is taken into account, such recursive call is no longer possible due to the need of binding variable \texttt{Y} of sort \texttt{NzNeg} to an expression opposite(\texttt{Y}) of sort \texttt{NzPos}, which is not possible in the (sub)sort hierarchy given by \texttt{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:

\begin{itemize}
  \item [\texttt{http://zenon.dsic.upv.es/muterm}]
\end{itemize}

\begin{itemize}
  \item Partially supported by EU (FEDER) and MICINN grant TIN 2007-68093-C02-02.
\end{itemize}
MU-TERM [23, 2] was originally designed to prove termination of Context-Sensitive Rewriting (CSR, [21]), where reductions are allowed only for specific arguments \( \mu(f) \subseteq \{1, \ldots, k\} \) of the \( k \)-ary function symbols \( f \) in the TRS. In this paper we report on the new features included in MU-TERM 5.0, not only to improve its ability to prove termination of CSR but also to verify a number of other termination properties of (variants of) TRSs.

In contrast to transformational approaches which translate termination problems into a classical termination problem for TRSs, we have developed specific techniques to deal with termination of CSR, innermost CSR, order-sorted rewriting and rewriting modulo specific axioms (associative or commutative) by using dependency pairs (DPs, [7]). Our benchmarks show that direct methods lead to simpler, faster and more successful proofs. Moreover, MU-TERM 5.0 has been rewritten to embrace the dependency pair framework [17], a recent formulation of the dependency pair approach which is specially well-suited for mechanizing proofs of termination.

2 Structure and Functionality of MU-TERM 5.0

MU-TERM 5.0 consists of 47 Haskell modules with more than 19000 lines of code. A web-based interface and compiled versions in several platforms are available at the MU-TERM 5.0 web site. In the following, we describe its new functionalities.

2.1 Proving Termination of Context-Sensitive Rewriting

As in the unrestricted case [7], the context-sensitive dependency pairs (CSDPs, [3]) are intended to capture all possible function calls in infinite \( \mu \)-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\(^1\) (TPDB) which contains 109 examples\(^2\).

<table>
<thead>
<tr>
<th>Termination Tool</th>
<th>Total</th>
<th>Yes</th>
<th>No</th>
<th>CSDP(s)</th>
<th>CSRPO</th>
<th>Transf.</th>
<th>Average (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>MU-TERM 5.0</td>
<td>99/109</td>
<td>95</td>
<td>4</td>
<td>59</td>
<td>0</td>
<td>0</td>
<td>0.95s</td>
</tr>
<tr>
<td>MU-TERM 4.3</td>
<td>64/109</td>
<td>64</td>
<td>0</td>
<td>54</td>
<td>7</td>
<td>3</td>
<td>3.44s</td>
</tr>
</tbody>
</table>

Table 1. MU-TERM 4.3 compared to MU-TERM 5.0 in proving termination of CSR.

The results show the power of the new CSDP framework in MU-TERM 5.0, not only by solving more examples in less time, but also disregarding the need of using transformations or CSRPO for solving them.

### 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 \textit{eager} programming languages like Maude and OBJ\(^3\) 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 \((R, \mu)\) is transformed into a TRS whose innermost termination implies the innermost termination for \((R, \mu)\). 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

\(1\) See \url{http://termination-portal.org/wiki/TPDB}

\(2\) 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 \cite{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 CSDF framework \cite{1}. Under some conditions, termination of CSR and termination of innermost CSR coincide \cite{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 \cite{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\footnote{The Maude Termination Tool \cite{12} implements a number of transformations from OS-TRSs into TRSs which can also be used for this purpose.}. 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\lor C-Rewriting

Recently, we have developed a suitable dependency pair framework for proving termination of A\lor C-rewrite theories \cite{5}. An A\lor 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 \cite{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 modulo axioms: CoME \cite{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 \cite{24}.
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 $\mathbb{N}^2 = \{0, 1, 2\}$.
   (d) RT processor with LPoly and coefficients in $\mathbb{Q}^2 = \{0, 1, 2, \frac{1}{2}\}$ (in this order).
   (e) RT processor with simple mixed polynomials (SMPoly) and coefficients in $\mathbb{N}^2$.
   (f) RT processor with SMPoly and rational coefficients in $\mathbb{Q}^2$.
   (g) RT processor with 2-square matrices with entries in $\mathbb{N}^2$ and $\mathbb{Q}^2$.
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 (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

\[http://www.lcc.uma.es/~duran/MTT\]
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 Tools⁵. Moreover, in the standard subcategory, we have obtained quite good results in the 2009 and 2010 editions and in the equational category in 2010.

Note also that MU-TERM 5.0 has a web interface that allows inexpert users to prove automatically termination by means of the ‘automatic’ option. This is very convenient for teaching purposes, for instance. And, apart from MTT, it is the only termination tool that accepts programs in OBJ/Maude syntax.

Therefore, MU-TERM 5.0 is no more a tool for proving termination of CSR only. We can say now that it has evolved to become a powerful termination tool which is able to prove termination of a wide range of interesting properties

⁵ See http://www.lri.fr/~marche/termination-competition/2007/, where only AProVE and MU-TERM participated, and http://termcomp.uibk.ac.at/termcomp/ where there were three more tools in the competition: AProVE, Jambox [13] (only in the 2009 edition), and VMTL [27]. AProVE and MU-TERM solved the same number of examples but MU-TERM was much faster. The 2008 edition had only one participant: AProVE.
of rewriting with important applications to prove termination of programs in sophisticated rewriting-based programming languages like Maude or OBJ.*

References


Extended Versions

8.14 Proving Termination in the Context-Sensitive Dependency Pair Framework

Proving Termination in the Context-Sensitive Dependency Pair Framework

Raúl Gutiérrez1 and Salvador Lucas1
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 \( \mu(f) \subseteq \{1, \ldots, \omega(f)\} \) for every function symbol \( f \) of arity \( \omega(f) \) in the signature \( F \) 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.
Example 1. Consider the following TRS in [1]:

\[
\begin{align*}
gt(0, y) &\rightarrow \text{false} \\
gt(x, 0) &\rightarrow \text{true} \\
gt(s(x), s(y)) &\rightarrow gt(x, y) \\
\text{minus}(z, y) &\rightarrow if(gt(y, z), \text{minus}(z, y), y) \\
\text{if}(true, x, y) &\rightarrow x \\
\text{div}(0, s(y)) &\rightarrow 0 \\
\text{div}(s(x), s(y)) &\rightarrow s(\text{div}(\text{minus}(x, y), s(y)))
\end{align*}
\]

with \(\mu(\text{if}) = \{1\}\) and \(\mu(f) = \{1, \ldots, \text{ar}(f)\}\) 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:

\[
\text{minus}(0, 0) \rightarrow false \text{ if(gt(0, 0), minus(0, 0), 0)} \rightarrow ... 
\]

In CSR, though, this sequence is not possible because reductions on 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 \(R\) 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\(^1\) \(g\). 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 \(u = \ell(t_1, \ldots, t_k)\) if \(\ell = f(t_1, \ldots, t_k)\) and \(v = g(t_1, \ldots, t_n)\) if \(s = g(t_1, \ldots, t_n)\) is a subterm of \(r\) and \(g\) is a defined symbol. The notation \(\ell f\) for a given symbol \(f\) means that \(f\) is marked. In practice, we often capitalize \(f\) and use \(F\) instead of \(\ell f\) in our examples. For this reason, the dependency pair technique starts by considering a new TRS \(\text{DP}(R)\) which contains all these dependency pairs for each \(\ell \rightarrow r\) in \(R\). The rules in \(R\) and \(\text{DP}(R)\) determine the so-called dependency chains whose finiteness characterizes termination of \(R\) [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 \(\ell f \rightarrow s f^\ast\) 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., \(u = \ell f\) 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

\(^1\) A symbol \(g \in F\) is defined in \(R\) if there is a rule \(\ell \rightarrow r\) in \(R\) whose left-hand side \(\ell\) is of the form \(g(t_1, \ldots, t_k)\) for some \(k \geq 0\).
the non-$\mu$-replacing parts of the terms involved in the infinite sequence until a \textit{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.

\textbf{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):

\begin{align*}
\text{GT}(s(x), s(y)) & \rightarrow \text{GT}(x, y) \quad (1) \\
\text{M}(x, y) & \rightarrow \text{IF}(y, 0, \text{minus}(p(x), p(y)), x) \quad (7) \\
\text{D}(s(x), s(y)) & \rightarrow \text{D}(\text{min}(x, y), s(y)) \quad (8) \\
\text{IF}(\text{true}, x, y) & \rightarrow U(x) \quad (4) \\
\text{IF}(\text{false}, x, y) & \rightarrow U(x) \quad (5) \\
\text{U}(p(x)) & \rightarrow \text{P}(x) \quad (6) \\
\text{U}(\text{min}(x, y)) & \rightarrow \text{U}(x) \quad (12) \\
\text{U}(\text{min}(x, y)) & \rightarrow \text{U}(y) \quad (13)
\end{align*}

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:

\begin{align*}
\text{GT}(s(x), s(y)) & \rightarrow \text{GT}(x, y) \quad (1) \\
\text{M}(x, y) & \rightarrow \text{IF}(y, 0, \text{minus}(p(x), p(y)), x) \quad (7) \\
\text{D}(s(x), s(y)) & \rightarrow \text{D}(\text{min}(x, y), s(y)) \quad (8) \\
\text{IF}(\text{true}, x, y) & \rightarrow x \quad (14) \\
\text{IF}(\text{false}, x, y) & \rightarrow y \quad (15)
\end{align*}

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
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 \(p, q, \ldots\) are represented by chains of positive natural numbers used to address subterms of \(t\). Given positions \(p, q\), we denote its concatenation as \(pq\). If \(p\) is a position, and \(Q\) is a set of positions, then \(p,Q = \{pq \mid q \in Q\}\). We denote the root or top position by \(\Lambda\). The set of positions of a term \(t\) is \(\text{Pos}(t)\). Positions of nonvariable symbols \(f \in F\) in \(t \in T(F,X)\) are denoted as \(\text{Pos}_F(t)\). The subterm at position \(p\) of \(t\) is denoted as \(t_p\), and \(t|_p\) is the term \(t\) with the subterm at position \(p\) replaced by \(s\). We write \(t ⊵ s\) if \(s = t_p\) for some \(p \in \text{Pos}(t)\) and \(t > s\) if \(t ⊵ s\) and \(t \neq s\). The symbol labeling the root of \(t\) is denoted as \(\text{root}(t)\). A substitution is a mapping \(\sigma : X \rightarrow T(F,X)\) from a set of variables \(X\) into the set \(T(F,X)\) of terms built from the symbols in the signature \(F\) and the variables \(X\). A context is a term \(C \in T(F' \cup \{\Box\}, X')\) with a 'hole' \(\Box\) (a fresh constant symbol). A rewrite rule is an ordered pair \((\ell, r)\), written \(\ell \rightarrow r\), with \(\ell, r \in T(F,X)\), \(\ell \not\in X\) 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 \(R = (F, R)\) where \(F\) is a signature and \(R\) is a set of rewrite rules over terms in \(T(F,X)\). Given \(\mathcal{R} = (F, R)\), we consider \(F\) as the disjoint union \(F = C \cup D\) of symbols \(c \in C\), called constructors and symbols \(f \in D\), called defined symbols, where \(D = \{\text{root}(\ell) \mid \ell \rightarrow r \in R\}\) and \(C = F \setminus D\).

In the following, we introduce some notions and notation about CSR [19]. A mapping \(\mu : F \rightarrow \mathbb{N}\) is a replacement map if \(\forall f \in F, \mu(f) \subseteq \{1, \ldots, \omega(f)\}\). Let \(M_\mu\) be the set of all replacement maps (or \(M_\mu\) for the replacement maps of a TRS \(R = (F, R)\)). The set of \(\mu\)-replacing positions \(\text{Pos}^\mu(t)\) of \(t \in T(F,X)\) is: \(\text{Pos}^\mu(t) = \{\ell \mid t \in X\} \cup \bigcup_{f \in \text{Pos}(\ell)} \mu(f) \cdot \text{Pos}^\mu(f)\), if \(t \not\in X\). The set of \(\mu\)-replacing variables of \(t\) is \(\text{Var}^\mu(t) = \{x \in \text{Var}(t) \mid 3p \in \text{Pos}(t) \\setminus \text{Pos}^\mu(t), \mu(p) = x\}\) and \(\text{Var}^\mu(t) = \{x \in \text{Var}(t) \mid 3p \in \text{Pos}(t) \\setminus \text{Pos}^\mu(t), \mu(p) = x\}\) is the set of non-\(\mu\)-replacing variables of \(t\). Note that \(\text{Var}^\mu(t)\) and \(\text{Var}^\mu(t)\) do not need to be disjoint. The \(\mu\)-replacing subterm relation \(\triangleright_\mu\) is given by \(t \triangleright_\mu s\) if there is \(p \in \text{Pos}^\mu(t)\) such that \(s = t|_p\). We write \(t \triangleright_\mu s\) if \(t \triangleright_\mu s\) and \(t \neq s\). We write...
\( s △ µ \) \( s \) to denote that \( s \) is a non-\( µ \)-replacing strict subterm of \( t \), i.e., there is a non-\( µ \)-replacing position \( p \in \text{Pos}(t) \setminus \text{Pos}^0(t) \) such that \( s = t[p] \). In CSR, we (only) contract \( µ \)-replacing redexes: \( t \) \( µ \)-rewrites to \( s \), written \( t \to_{\text{r}µ} s \) (or \( t \to_{\text{r}µ} s \) to make position \( p \) explicit), if there are \( ℓ \to r \in R \), \( p \in \text{Pos}^0(t) \) and a substitution \( σ \) such that \( t[p] = σ(t) \) and \( s = t[σ(r)] \). \( t \to_{\text{r}µ} s \) means that the \( µ \)-rewrite step is applied below position \( q \), i.e., \( p > q \). We say that a variable \( x \) is migrating in \( ℓ \to r \in R \) if \( x \in \text{Var}^ω(ℓ) \setminus \text{Var}^ω(r) \). A term \( t \) is \( µ \)-terminating if there is no infinite \( µ \)-rewrite sequence \( t = t_1 \to_{\text{r}µ} t_2 \to_{\text{r}µ} \cdots \to_{\text{r}µ} t \). A TRS \( R = (F, R) \) is \( µ \)-terminating if \( (F, R) \) is \( µ \)-terminating. A pair \( (R, µ) \) where \( R \) is a TRS and \( µ \in M_R \) is often called a \( CS-TRS \).

3 Infinite \( µ \)-Rewrite Sequences

Let \( M_{∞, µ} \) be a set of minimal non-\( µ \)-terminating terms in the following sense: \( t \) belongs to \( M_{∞, µ} \) if \( t \) is non-\( µ \)-terminating and every strict \( µ \)-replacing subterm \( s \) of \( t \) (i.e., \( t △ µ s \) \( s \) is \( µ \)-terminating [2]. Minimal terms allow us to characterize infinite \( µ \)-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 \( σ(x) \) 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 \( T_{∞, µ} \) be a set of strongly minimal non-\( µ \)-terminating terms in the following sense: \( t \) belongs to \( T_{∞, µ} \) if \( t \) is non-\( µ \)-terminating and every strict subterm \( u \) (i.e., \( t △ u \)) is \( µ \)-terminating. It is obvious that \( \text{root}(t) \in D \) for all \( t \in T_{∞, µ} \).

Every non-\( µ \)-terminating term has a subterm that is strongly minimal. Then, given a non-\( µ \)-terminating term \( t \) we can always find a subterm \( t_0 \in T_{∞, µ} \) of \( t \) which starts a minimal infinite \( µ \)-rewrite sequence of the form \( t_0 \to_{A_1} \to_{A_2} \cdots \to_{A_∞} t \), where \( t_0, σ(t_i) \in M_{∞, µ} \) for all \( i > 0 \) [3]. Theorem 1 below tells us that we have two possibilities:

- The minimal non-\( µ \)-terminating terms \( t_i \in M_{∞, µ} \) in the sequence are partially introduced by a \( µ \)-replacing nonvariable subterm of the right-hand sides \( r_i \) of the rules \( ℓ_i \to r_i \).
- The minimal non-\( µ \)-terminating terms \( t_i \in M_{∞, µ} \) in the sequence are introduced by instantiated migrating variables \( x_i \) of (the respective) rules \( ℓ_i \to r_i \), i.e., \( x_i \in \text{Var}^ω(r_i) \setminus \text{Var}^ω(ℓ_i) \). Then, \( t_i \) is partially introduced by terms occurring at non-\( µ \)-replacing positions in the right-hand sides of the rules (hidden terms) within a given (hiding) context.

We use the following functions [2, 3]: \( \text{Ren}^ω(t) \), which independently renames all occurrences of \( µ \)-replacing variables by using new fresh variables which are not
in \( \text{Var}(t) \), and \( \text{Narr}_\mu(t) \), which indicates whether \( t \) is \( \mu \)-narrowable\(^2\) (w.r.t. the intended TRS \( R \)).

A nonvariable term \( t \in T(\mathcal{F}, \mathcal{X}) \setminus \mathcal{X} \) 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, \( \mathcal{H}(R, \mu) \) is the set of all hidden terms in \((R, \mu)\) and \( \mathcal{H}(R, \mu) \) the set of \( \mu \)-narrowable hidden terms headed by a defined symbol:

\[
\mathcal{H}(R, \mu) = \{ t \in \mathcal{H}(R, \mu) \mid \text{root}(t) \in D \text{ and } \text{Narr}_\mu(t) \}
\]

**Definition 2 (Hiding Context).** Let \( R \) be a TRS and \( \mu \in M_R \). A function symbol \( f \) hides position \( i \) in the rule \( \ell \rightarrow r \in R \) if \( f \neq f(r_1, \ldots, r_n) \) for some terms \( r_1, \ldots, r_n \) and there is \( i \in f(r) \) such that \( f_r \) contains a \( \mu \)-replacing defined symbol (i.e., \( \text{Pos}_\mu(f_r) \neq \emptyset \)) or a variable \( x \in \text{Var}_\mu(f_r) \) \( \cup \text{Var}_\mu(r) \) which is \( \mu \)-replacing in \( r_1 \) (i.e., \( x \in \text{Var}_\mu(r_1) \)). A context \( C[i] \) is hiding \([i]\) if \( C[i] = \emptyset \), or \( C[i] = \ell(t_1, \ldots, t_{k-1}, C'[\emptyset], t_k, \ldots, t_n) \), where \( \ell \) hides position \( i \) and \( C'[\emptyset] \) is a hiding context.

**Definition 2** is a refinement of \([1, \text{Definition 7}]\), 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 1.** The hidden terms in Example 1 are \( \mu \)-narrowable terms headed by a defined symbol:

\[\text{os}_1 \in \text{os}(\ell(x, y)) \]$\text{os}_2 \in \text{os}(\ell(\mu(x)))$\text{os}_3 \in \text{os}(\ell(\mu(y)))$\mu_1 \in \text{os}(\ell(\mu(x))))$

Symbol \( \mu \) hides positions 1 and 2, but \( \mu \) hides no position. Without the new condition in Definition 2, \( \mu \) 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 \( R \) be a TRS and \( \mu \in M_R \). For all \( t \in \mathcal{T}_{\mathcal{X}, \mu} \), there is an infinite sequence:

\[
t = t_0 \xrightarrow{\Delta_{\mathcal{X}, \mu}} \sigma_1(t_1) \xrightarrow{\Delta_{\mathcal{X}, \mu}} \sigma_1(t_1) \xrightarrow{\Delta_{\mathcal{X}, \mu}} \sigma_2(t_2) \xrightarrow{\Delta_{\mathcal{X}, \mu}} \cdots
\]

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

1. \( t_i = \sigma_i(s_i) \) for some nonvariable term \( s_i \), such that \( r_i \xrightarrow{\mu} s_i \), or
2. \( \sigma_i(s_i) = \theta_i(C'_i[t'_i]) \) and \( t_i = \theta_i(t'_i) \) for some variable \( x_i \in \text{Var}_\mu(r_i) \setminus \text{Var}_\mu(\ell_i) \), \( C'_i[t'_i] \in \mathcal{H}(R, \mu) \), hiding context \( C'_i[\emptyset] \), 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.

\(^2\)A term \( s \) \( \mu \)-narrow to the term \( t \) if there is a nonvariable position \( p \in \text{Pos}_\mu(s) \) and a rule \( \ell \rightarrow r \) such that \( s_p \) and \( t \) unify with \( s_p[r]_\mu \).
Definition 3 (Context-Sensitive Dependency Pairs [3]). Let \( R = (F, R) \) be a TRS and \( \mu \in M_F \). We define \( \text{DP}(R, \mu) \) to be set of context-sensitive dependency pairs (CSDPs) where:

\[
\text{DP}_F(R, \mu) = \{ \ell \to s^\nu \mid \ell \to r \in R, r \not\in_\mu s, \text{root}(s) \in D, \ell \not\in_\mu s, \text{NARB}^\mu (\text{Ren}(s)) \}
\]

\[
\text{DP}_A(R, \mu) = \{ \ell \to x \mid \ell \to r \in R, x \in \text{Var}(r) \setminus \text{Var}(\ell) \}
\]

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

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 \( R \) be a TRS and \( \mu \in M_R \). We define \( \text{unh}(R, \mu) \) as the TRS consisting of the following rules:

1. \( f(x_1, \ldots, x_k) \to x_i \) for all function symbols \( f \) of arity \( k \), distinct variables \( x_1, \ldots, x_k \), and \( 1 \leq i \leq k \) such that \( f \) hides position \( i \) in \( \ell \to r \in R \), and

2. \( t \to \ell^\sharp \) for every \( t \in \text{NHT}(R, \mu) \).

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

\[
\text{minus}(p(x), p(y)) \to M(p(x), p(y)) \quad \text{minus}(x, y) \to y \\
p(x) \to P(x) \quad \text{minus}(x, y) \to x
\]

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 \( S \), we let \( S_{\mu, \rho} \) be the rules from \( S \) of the form \( s \to i \in S \) and \( s \not\in_\mu t \); and \( S = S \setminus S_{\mu, \rho} \).

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

1. if \( v_i \not\in \text{Var}(u_i) \setminus \text{Var}^\rho(u_i) \), then \( \sigma(v_i) = t_i \to^\rho_{\mu, \rho} \sigma(u_{i+1}) \), and

2. if \( v_i \in \text{Var}(u_i) \setminus \text{Var}^\rho(u_i) \), then \( \sigma(v_i) = t_i \to^S_{\mu, \rho} \circ \to^A_{\mu, \rho} t_i \to^R_{\mu, \rho} \sigma(u_{i+1}) \).

A \((P, R, S, \mu, \rho)\)-chain is called minimal if for all \( i \geq 1 \), \( t_i \) is \((R, \mu)\)-terminating.

Notice that if rules \( f(x_1, \ldots, x_k) \to x_i \) for all \( f \in D \) and \( i \in \mu^\sharp(f) \) (where \( x_1, \ldots, x_k \) are variables) are used in Item 1 of Definition 4, then Definition 5 yields the notion of chain in [3]; and if, additionally, rules \( f(x_1, \ldots, x_k) \to f'(x_1, \ldots, x_k) \) for all \( f \in D \) 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 \( R \) be a TRS and \( \mu \in M_R \). A CS-TRS \((R, \mu)\) is terminating if and only if there is no infinite \( \text{DP}(R, \mu) \cup \text{unh}(R, \mu, \mu^\sharp) \)-chain.
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 \( \mu \in M_{\mathcal{P}, \mathcal{R}, \mathcal{S}} \). 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 \( \text{Proc}(\tau) \neq \emptyset \) 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 \( \text{Proc}(\tau) \neq \emptyset \) and \( \exists \tau' \in \text{Proc}(\tau) \) where \( \tau' \) 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 \((\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^\prime)\). For every inner node labeled with \( \tau \), there is a sound processor \( \text{Proc} \) satisfying one of the following conditions:

1. \( \text{Proc}(\tau) = \emptyset \) 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. \( \text{Proc}(\tau) \neq \emptyset \), \( \exists \tau' \in \text{Proc}(\tau) \) 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 [3] into our framework. The pairs in a CS-TRS \((\mathcal{P}, \mu)\), where \( \mathcal{P} = (\mathcal{G}, \mathcal{P}) \), are partitioned as follows: \( \mathcal{P}_X = \{ u \rightarrow v \in \mathcal{P} \mid v \in \text{Var}(u) \setminus \text{Var}(u) \} \) and \( \mathcal{P}_Y = \mathcal{P} \setminus \mathcal{P}_X \).

**Theorem 4 (Collapsing Pair Transformation).** Let \( \tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu) \) be a CS problem where \( \mathcal{P} = (\mathcal{G}, \mathcal{P}) \) and \( \mathcal{P}_0 \) be given by the following rules:

- \( u \rightarrow U(x) \) for every \( u \rightarrow x \in \mathcal{P}_X \),
- \( U(s) \rightarrow U(t) \) for every \( s \rightarrow t \in \mathcal{S}_{\mathcal{P}_X} \), and
Proofing Termination in the CSDP Framework

• $U(s) \rightarrow t$ for every $s \rightarrow t \in S^\#$.

Here, $U$ is a new fresh symbol. Let $P' = (P \setminus P_U) \cup P_U$, and $\mu'$ extends $\mu$ by $\mu'(U) = \emptyset$. The processor $\text{Proc}_{\text{Coll1}}$ given by $\text{Proc}_{\text{Coll1}}(\tau) = \{(P', \mathcal{R}, \emptyset, \mu')\}$ 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 $\text{mu-term}$ [4]. The subTRS of $P_U$ containing the rules whose migrating variables occur on non-$\mu$-replacing immediate subterms in the left-hand side is $P^*_U = \{(f(u_1, \ldots, u_k) \rightarrow x \in P^*_A \mid \exists i. 1 \leq i \leq k, i \notin \mu(f), x \in \text{Var}(u_i)\}$.

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

\[
\text{Proc}_{\text{Coll1}}(\tau) = \begin{cases} 
\emptyset & \text{if } \mathcal{P} = P^*_U \\
\{(P, \mathcal{R}, S, \mu)\} & \text{otherwise}
\end{cases}
\]

is sound and complete.

Example 5. (Continuing Example 1) Consider the CS problem $\tau = (P_4, R, S_4, \mu^4)$ where $P_4 = \{14, 15\}$ and $S_4 = \{16, 18, 19\}$. We can apply $\text{Proc}_{\text{Coll1}}(\tau)$ 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 $R$. The nodes of the graph are the dependency pairs in $\mathcal{DP}(R)$ and there is an arc from a dependency pair $u \rightarrow v$ to a dependency pair $v' \rightarrow v''$ if there are substitutions $\theta$ and $\theta'$ such that $\theta(v) \rightarrow_{R}^* \theta'(v'')$. In our setting, we have the following.

Definition 7 (Context-Sensitive Graph of Pairs). Let $R$, $P$ and $S$ be TRSs and $\mu \in \text{Mu}(P, S)$ and $\mu \in \text{Mu}(P, S)$. The context-sensitive (CS) graph $G(P, R, S, \mu)$ has $P$ as the set of nodes. Given $u \rightarrow v, u' \rightarrow v' \in P$, there is an arc from $u \rightarrow v$ to $u' \rightarrow v'$ if $u \rightarrow v, u' \rightarrow v'$ is a minimal $(P, R, S, \mu)$-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.
Theorem 6 (SCC Processor). Let \( \tau = (P, R, S, \mu) \) be a CS problem. Then, the processor \( \text{Proc}_{SCC} \) given by

\[
\text{Proc}_{SCC}(\tau) = \{(Q, R, S_Q, \mu) \mid \text{Q contains the pairs of an SCC in } G(P, R, S, \mu) \}
\]

where

- \( S_Q = \emptyset \) if \( Q_X = \emptyset \).

Theorem 7 (Approximation of the CS Graph). Let \( \tau = (P, R, S, \mu) \) be a CS problem. The estimated CS graph

\[
\text{EG} = \{ (u, v) \mid \text{there is an arc from } u \rightarrow v \in P \}
\]

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 \( \text{TCap}_R \) [3], which renames all subterms headed by a ‘defined’ symbol in \( R \) by new fresh variables if it can be rewritten:

Definition 8 (TCap\(_R^p\) [3]). Given a TRS \( R \) and a replacement map \( \mu \), we let \( \text{TCap}_R^p \) be as follows:

\[
\text{TCap}_R^p(x) = y \quad \text{if } x \text{ is a variable, and}
\]

\[
\text{TCap}_R^p(f(t_1, \ldots, t_k)) = \begin{cases} 
\ell & \text{if there is } \ell \rightarrow r \text{ in } R \\
y & \text{otherwise}
\end{cases}
\]

where \( y \) is a new fresh variable, \( y_i^p = \text{TCap}_R^p(s) \) if \( i \in \mu(f) \) and \( y_i^p = s \) if \( i \notin \mu(f) \). We assume that \( \ell \) shares no variable with \( f(t_1, \ldots, t_k) \) when the unification is attempted.

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

1. there is an arc from \( u \rightarrow v \in P \) to \( u' \rightarrow v' \in P \) if \( \text{TCap}_R^p(v) \) and \( u' \) unify, and
2. there is an arc from \( u \rightarrow v \in P \) to \( u' \rightarrow v' \in P \) if there is \( s \rightarrow t \in S \) such that \( \text{TCap}_R^p(t) \) and \( u' \) unify.

We have the following.

Theorem 8 (SCC Processor using TCap\(_R^p\)). Let \( \tau = (P, R, S, \mu) \) be a CS problem. The CS processor \( \text{Proc}_{SCC} \) given by

\[
\text{Proc}_{SCC}(\tau) = \{(Q, R, S_Q, \mu) \mid \text{Q contains the pairs of an SCC in } \text{EG}(P, R, S, \mu) \}
\]

where

- \( S_Q = \emptyset \) if \( Q_X = \emptyset \).
is sound and complete.

Example 6. In Figure 1 (right) we show $EG(DP(R, \mu), R, unh(R, \mu), \mu^2)$ for $R$ in Example 1. The graph has three SCCs $P_1 = \{(1)\}$, $P_2 = \{(8)\}$, and $P_3 = \{(7), (14), (15)\}$. If we apply the CS processor $Proc_{CS}$ to the initial CS problem $(DP(R, \mu), R, unh(R, \mu), \mu^2)$ for $(R, \mu)$ in Example 1, then we obtain the problems: $(P_1, R, \varnothing, \mu^2)$, $(P_2, R, \varnothing, \mu^2)$, $(P_3, R, S_3, \mu^2)$, where $S_3 = \{(16), (18), (19)\}$.

5.3 Reduction Triple Processor

A $\mu$-reduction pair $(\geq, \sqsubseteq)$ consists of a stable and $\mu$-monotonic quasi-ordering $\geq$, and a well-founded stable relation $\sqsubseteq$ on terms in $T(\mathcal{F}, \mathcal{A})$ which are compatible, i.e., $\geq \circ \sqsubseteq \sqsubseteq$ or $\sqsubseteq \circ \geq \sqsubseteq [2]$.

In [2, 3], when a collapsing pair $u \rightarrow z$ occurs in a chain, we have to look inside the instantiated right-hand side $\sigma(z)$ 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 $\geq$ of reduction pairs $(\geq, \sqsubseteq)$ which are used in [2, 3] are required to have the $\mu$-subterm property, i.e., $\geq \circ \sqsubseteq \sqsubseteq [2]$. This is equivalent to impose $f(x_1, \ldots, x_k) \geq x_i$ for all projection rules $f(x_1, \ldots, x_k) \rightarrow x_i$, with $f \in \mathcal{F}$ and $i \in \mu(f)$. This is similar for markings: in [2] we have to ensure that $f(x_1, \ldots, x_k) \geq f(x_1, \ldots, x_k)$ for all defined symbols $f$ in the signature. In [3], thanks to the notion of hidden term, we relaxed the last condition: we require $t \geq t'$ for all (narrowable) hidden terms $t$. In [1], thanks to the notion of hiding context, we only require that $\geq$ is compatible with the projections $f(x_1, \ldots, x_k) \rightarrow x_i$ for those symbols $f$ and positions $i$ such that $f$ hides position $i$. However, this information is implicitly encoded as (new) pairs $U(f(x_1, \ldots, x_k)) \rightarrow U(x_i)$ in the set $P$. The strict component $\sqsubseteq$ of the reduction pair $(\geq, \sqsubseteq)$ is used with these new pairs now.

In this paper, since the rules in $S$ 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 $S$ 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 $(\geq, \sqsubseteq, \triangleright)$ now, where $(\geq, \sqsubseteq)$ is a $\mu$-reduction pair and $\triangleright$ is a stable relation on terms which is compatible with $\geq$ or $\sqsubseteq$, i.e., $\geq \circ \triangleright \sqsubseteq \geq$ or $\sqsubseteq \circ \triangleright \sqsubseteq \sqsubseteq$.

Theorem 9 ($\mu$-Reduction Triple Processor). Let $\tau = (\mathcal{P}, R, S, \mu)$ be a CS problem. Let $(\geq, \sqsubseteq, \triangleright)$ be a $\mu$-reduction triple such that

- $S_{\triangleright} = S_{\triangleright} \cup \{s \rightarrow t \mid s \rightarrow t \in S_{\triangleright}, TCap^\mu_R(t) \text{ and } u' \text{ unify for some } u' \rightarrow v' \in Q\}$ if $Q \neq \emptyset$.

$\rightarrow$
1. \( P \subseteq \geq \cup \bowtie \geq \), \( R \subseteq \geq \), and
2. whenever \( P_X \neq \emptyset \) we have that \( S \subseteq \geq \cup \bowtie \cup \trianglerighteq \).

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

\[
\text{Proc}_{RT}(\tau) = \begin{cases} 
\{ \{(P \setminus P_\bowtie, R, S \setminus S_\bowtie, \mu) \} \} & \text{if (1) and (2) hold} \\
\{(P, R, S, \mu)\} & \text{otherwise}
\end{cases}
\]

is sound and complete.

Since rules from \( S \) are only applied after using a collapsing pair, we only need to make them compatible with some component of the triple if \( P \) contains collapsing pairs, i.e., if \( P_X \neq \emptyset \). Another advantage is that we can now remove rules from \( S \).

5.4 Reduction Triple Processor with Usable Rules

In order to use \( \text{Proc}_{RT} \), we require that the rules in the TRS \( R \) of the CS problem \( (P, R, S, \mu) \) are included in \( \geq \), i.e., \( R \subseteq \geq \) 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 \( R \) 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 \( Rls_R(f) = \{ \ell \rightarrow r \in R \mid \text{root}(\ell) = f \} \). The set of \( \mu \)-replacing symbols in a term \( t \in T(F, X) \) is denoted by \( Fun^\mu(t) = \{ f \mid 3p \in Pos^\mu(t), f = \text{root}(t_p) \} \). The simplest adaptation of this notion is the following.

**Definition 10 (Basic \( \mu \)-Dependency \([16]\)).** Given a TRS \( (F, R) \) and \( \mu \in M_F \), we say that \( f \in F \) has a basic \( \mu \)-dependency on \( h \in F \) (written \( f \triangleleft_{\mu, h} \)) if \( f = h \) or there is a function symbol \( g \) with \( g \triangleleft_{\mu, h} \) and a rule \( \ell \rightarrow r \in Rls_R(f) \) with \( g \in Fun^\mu(\ell) \).

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

**Definition 11 (Basic CS Usable Rules).** Let \( \tau = (P, R, S, \mu) \) be a CS problem. The set \( U^\tau(\tau) \) of basic context-sensitive usable rules of \( \tau \) is

\[
U^\tau(\tau) = \bigcup_{u \rightarrow v \in P, f \in Fun^\mu(v) \triangleleft_{\mu, g}} Rls_R(g)
\]

However, Definition 11 does not lead to a correct approach for proving termination of CSR.
Example 7. Consider the TRS $R[5]$:

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

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

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

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

$$b \rightarrow B$$

Since there is no collapsing pair in $DP(R, \mu)$, after applying $\text{Proc}_{SCC}$ to $\tau_0 = (DP(R, \mu), R, \text{unh}(R, \mu), \mu)$, we obtain $
\tau_1 = (DP(R, \mu), R, \emptyset, \mu)$.

According to Definition 11, we have no basic usable rules for $\tau$ because $F(x, x)$ contains no symbol in $F$. We could wrongly conclude finiteness of $\tau_1$ the CS problem and, hence, $\mu$-termination of $(R, \mu)$, but we have the following infinite minimal $(P, R, S, \mu)$-chain where $b \rightarrow c(b)$ is used:

$$G(a) \hookrightarrow \rightarrow P, \mu$$
$$F(g(b)) \hookrightarrow \rightarrow P, \mu$$
$$\Lambda \hookrightarrow \rightarrow S, \mu$$
$$G(b) \hookrightarrow \rightarrow R, \mu$$
$$G(a) \hookrightarrow \rightarrow P, \mu$$

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 $R$ be a TRS and $\mu \in M_R$. A rule $\ell \rightarrow r$ is strongly $\mu$-conservative if it is $\mu$-conservative and $\text{Var}^\ell(\ell) \cap \text{Var}^\ell(r) = \emptyset$; and $R$ is strongly $\mu$-conservative if all rules in $R$ 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 $(P, R, S, \mu)$ where $P$ is:

$$G(a) \rightarrow F(g(b))$$
$$F(x) \rightarrow x$$

the only rule in $R$ is:

$$b \rightarrow a$$

and $S$ 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 $R$ is usable because it is necessary to build the following infinite minimal $(P, R, S, \mu)$-chain:

$$G(a) \hookrightarrow_{P, \mu} F(g(b)) \hookrightarrow_{P, \mu} A \hookrightarrow_{S, \mu} G(b) \hookrightarrow_{R, \mu} G(a) \hookrightarrow_{P, \mu} \cdots$$
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 \(\mathcal{R}\):

\[
\begin{align*}
 a(x, y) &\rightarrow b(x, x) \\
 d(x, e) &\rightarrow a(x, x) \\
b(x, g) &\rightarrow d(x, x) \\
g &\rightarrow e
\end{align*}
\]

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:

\[
\begin{align*}
 A(x, y) &\rightarrow B(x, x) \\
 D(x, e) &\rightarrow A(x, x) \\
 B(x, g) &\rightarrow D(x, x)
\end{align*}
\]

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

\(\tau = (\text{DP}(\mathcal{R}, \mu), \mathcal{R}, \emptyset, \mu^4)\)

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 \((\mathcal{C}_\mu)\) rules:

\[
\begin{align*}
 c(x, y) &\rightarrow x \\
c(x, y) &\rightarrow y
\end{align*}
\]

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:

\[
A(g, g) \rightarrow_p B(g, g) \rightarrow_p D(g, g) \rightarrow_R D(g, e) \rightarrow_R A(g, g) \rightarrow_p \cdots
\]

because we have the following sequence using \(\mathcal{C}_\mu\)-rules instead:

\[
\begin{align*}
 A(c(g, e), c(g, e)) &\rightarrow_p B(c(g, e), c(g, e)) \rightarrow_{\mathcal{C}_\mu} B(c(g, e), g)) \rightarrow_p \\
 D(c(g, e), c(g, e)) &\rightarrow_{\mathcal{C}_\mu} D(c(g, e), e) \rightarrow_p A(c(g, e), c(g, e)) \rightarrow_p \cdots
\end{align*}
\]

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

\[
B(c(g, e), c(g, e)) \rightarrow_{\mathcal{C}_\mu} B(c(g, e), g))
\]

is not longer possible. Since the infinite sequence above can be regarded as an infinite \(\mu\)-rewrite sequence:

\[
\begin{align*}
 A(g, g) &\rightarrow_{p, \mu} B(g, g) \rightarrow_{p, \mu} D(g, g) \rightarrow_{\mathcal{R}, \mu} D(g, e) \rightarrow_{p, \mu} A(g, g) \rightarrow_{p, \mu} \cdots
\end{align*}
\]

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\(^4\). The set of non-\(\mu\)-replacing symbols in a term \(t \in \mathcal{T}(\mathcal{F}, \mathcal{X})\) is denoted by \(\mathcal{F}\text{un}\rho(t) = \{f \mid \exists p \in \text{Pos}(t) \setminus \text{Pos}^\rho(t), f = \text{root}(t_l)\}\).

\(^4\) A more detailed analysis can be found in [16]
Definition 13 (µ-Dependency [16]). Given a TRS \((F, R)\) and \(\mu \in M_F\), we say that \(f \in F\) has a \(\mu\)-dependency on \(h \in F\), written \(f \trianglerighteq_{\mu, R} h\), if \(f = h\) or there is a function symbol \(g\) with \(g \trianglerighteq_{\mu, R} h\) and a rule \(\ell \rightarrow r \in R\) with \(g \in Fun^w(\ell) \cup Fun(r)\).

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 \(\tau = (P, R, S, \mu)\) be a CS problem. The set \(U^\tau(\tau)\) of context-sensitive usable rules of \(\tau\) is
\[
U^\tau(\tau) = \bigcup_{u \to v \in P, f \in Fun(f(x)) : Fun(v) \cup Fun(r) \cup Fun(s) \cup Fun(t) \cup Fun(u)} Rls_R(g) \cup \bigcup_{u \to v \in P, f \in Fun(f(x)) : Fun(v) \cup Fun(r) \cup Fun(s) \cup Fun(t) \cup Fun(u)} Rls_R(g) \cup \bigcup_{u \to v \in P, f \in Fun(f(x)) : Fun(v) \cup Fun(r) \cup Fun(s) \cup Fun(t) \cup Fun(u)} Rls_R(g)
\]

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 \(C_\tau\)-compatibility of \(\preceq\), i.e., \(c(x, y) \succeq x\) and \(c(x, y) \succeq y\) holds for a fresh function symbol \(c\) [14].

Theorem 10 (µ-Reduction Triple Processor with Usable Rules). Let \(\tau = (P, R, S, \mu)\) be a CS problem. Let \((\preceq, \sqsubset, \succeq)\) be a \(\mu\)-reduction triple such that
1. \(P \subseteq \preceq \cup \sqsubset\),
2. at least one of the following holds:
   (a) \(U^\tau(\tau) \subseteq \preceq\), \(P \cup U^\tau(\tau)\) is strongly \(\mu\)-conservative, \(\succeq\) is \(C_\tau\)-compatible
   (b) \(U^\tau(\tau) \subseteq \succeq\), \(\sqsubset\) is \(C_\tau\)-compatible,
   (c) \(R \subseteq \succeq\).
3. and, whenever \(P_X \neq \emptyset\) we have that \(S \subseteq \preceq \cup \sqsubset \cup \succeq\).

Let \(P = \{u \rightarrow v \in P \mid u \sqsubset v\}\) and \(S = \{s \rightarrow t \in S \mid s \sqsubset t\}\). Then, the processor \(Proc_{UB}\) given by
\[
Proc_{UB}(\tau) = \begin{cases} \{(P \setminus P, R, S \setminus S, \mu)\} & \text{if (1), (2) and (3) hold} \\ \{(P, R, S, \mu)\} & \text{otherwise} \end{cases}
\]
is sound and complete.

Example 10. (Continuing Example 6) Consider the CS problem \(\tau = (P_3, R_3, S_3, \mu^3)\) where \(P_3 = \{7, 14, 15\}\), \(S_3 = \{16, 18, 19\}\) and \(R_3\) is the TRS in Example 1. If we apply \(Proc_{UB}\) to the CS problem \(\tau\) by using the \(\mu\)-reduction triple \((\preceq, \sqsubset, \succeq)\) where \(\preceq\) and \(\succeq\) are the orderings induced by the following polynomial interpretation (see [20] for missing notation and definitions):

\[
\begin{align*}
&\text{if}(x, y, z) = \frac{1}{2}x + y + z \\
&\text{min}(x, y) = 2x + 2 + \frac{1}{2} \\
&\phi(x) = \frac{1}{2}x \\
&\psi(x) = 2x + 2 \\
&\text{true} = 2 \\
&\text{false} = 0 \\
&\text{if}(x, y) = 2x + 2 + \frac{1}{2}
\end{align*}
\]
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
\[
\begin{align*}
|M(r,x,y)| \geq & \ |M(\text{div}(p(x), p(y)), x)| \\
|M(\text{div}(p(x), p(y)), x)| \geq & \ |\text{div}(p(x), p(y))| \geq |p(x)| \geq |x|.
\end{align*}
\]

Then, we get $\text{Proc}_{UR}(\tau) = \{(7), (15), \mathcal{R}, (16), \mu^3\}$.

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 $\mathcal{R} = (\mathcal{F}, \mathcal{R})$ be a TRS. The set of root symbols associated to $\mathcal{R}$ is:
\[
\text{Root}(\mathcal{R}) = \{\text{root}(\ell) \mid \ell \to \tau \in \mathcal{R} \} \cup \{\text{root}(\pi) \mid \ell \to \tau \in \mathcal{R}, \tau \notin \mathcal{X}\}
\]

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 $\ell$ in $\text{Root}(\mathcal{R})$ an argument position $i \in \{1, \ldots, k\}$. This mapping is extended to terms by
\[
\pi(t) = \begin{cases} 
\{i \mid \ell(f_1, \ldots, f_k) \text{ and } f \in \text{Root}(\mathcal{R})
\end{cases}
\]

\[
\begin{cases} 
\ell(t) & \text{if } \ell = f(t_1, \ldots, t_k) \text{ and } f \in \text{Root}(\mathcal{R}) \\
\ell & \text{otherwise}
\end{cases}
\]

Theorem 11 (Subterm Processor). Let $\tau = (\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)$ be a CS problem where $\mathcal{R} = (\mathcal{F}, \mathcal{R}) = (\mathcal{G} \cup \mathcal{D}, \mathcal{R})$, $\mathcal{P} = (\mathcal{G}, \mathcal{P})$ and $\mathcal{S} = (\mathcal{H}, \mathcal{S})$. Assume that (1) $\text{Root}(\mathcal{P}) \cap \mathcal{D} = \emptyset$, (2) the rules in $\mathcal{P}_0$ are noncollapsing, and (3) if $\mathcal{P}_X \neq \emptyset$, then for all $s \to t \in \mathcal{S}_y$, $\text{root}(t) \in \text{Root}(\mathcal{P})$. Let $\pi$ be a simple projection for $\mathcal{P}$. Let $\mathcal{S}_e = \{s \to t \mid s \to t \in \mathcal{S}_e\}$. Let $\mathcal{P}_e^{\mathcal{S}_e} = \{u \to v \mid u \in \mathcal{P} \mid \pi(u) \supseteq \pi(v)\}$ and $\mathcal{S}_e^{\mathcal{P}_e} = \mathcal{S}_e \cup \{s \to t \mid s \in \mathcal{S}_e \mid s \supseteq \pi(t)\}$. Then, $\text{Proc}_{\text{subterm}}$ given by
\[
\text{Proc}_{\text{subterm}}(\tau) = \begin{cases} 
\{(\mathcal{P} \setminus \mathcal{P}_e^{\mathcal{S}_e}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_e^{\mathcal{P}_e}, \mu) \} & \text{if } \pi(\mathcal{P}) \subseteq \mathcal{P}_e \text{ and } \forall \mathcal{P}_X \neq \emptyset, \\
\{(\mathcal{P}, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise}
\end{cases}
\]
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 \((P_0, R, S_0, \mu')\) where \(P_0 = \{(7), (15)\}\) and \(S_0 = \{(16)\}\). We can apply the subterm processor \(\text{Proc}_{\text{subterm}}\), by using the projection \(\pi(f) = 3\) and \(\pi(M) = 1\):

\[
\begin{align*}
\pi(M(x, y)) &= 2 \geq_M x = \pi(\text{IF}(y, 0, \text{minus}(p(x), p(y)), x)) \\
\pi(\text{IF}(\text{false, x, y})) &= y \geq_M y = \pi(y) \\
\pi(\text{minus}(p(x), p(y))) &= \text{minus}(p(x), p(y)) \geq_M p(x) = \pi(M(p(x), p(y)))
\end{align*}
\]

We obtain the CS problem \(\tau' = \{(7), (15)\}\) 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 \(R\), but only the rules in \(P \cup S\).

Theorem 12 (Non-\(\mu\)-Replacing Projection Processor). Let \(\tau = (P, R, S, \mu)\) be a CS problem where \(R = (C \cup D, R)\) and \(P = (G, P)\). Assume that (1) \(\text{Root}(P) \cap D = \emptyset\), (2) the rules in \(P_0\) are noncollapsing, and (3) if \(P_V \neq \emptyset\), then for all \(s \rightarrow t \in S_1\), \(\text{root}(t) \in \text{Root}(P)\). Let \(\pi\) be a simple projection for \(P\).

Let \(S_\pi = S_{\pi, \mu} \cup \{s \rightarrow \pi(t) \mid s \rightarrow t \in S_1\}\). Let \(\geq\) be a stable quasi-ordering on terms whose strict and stable part \(>\) is well-founded such that

1. for all \(f \in \text{Root}(P)\), \(\pi(f) \notin \mu(f)\),
2. \(\pi(P) \subseteq \geq\), and,
3. whenever \(S \neq \emptyset\) and \(P_V \neq \emptyset\), we have that \(S_\pi \subseteq \geq\).

Let \(P_{\pi, >} = \{u \rightarrow v \in P \mid \pi(u) > \pi(v)\}\) and \(S_{\pi, >} = \{s \rightarrow t \in S_{\pi, \mu} \mid s > t\} \cup \{s \rightarrow t \mid s \geq t\}\). Then, the processor \(\text{Proc}_{\text{NRP}}\) given by

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

is sound and complete.

Example 12. Consider the following TRS \(R\) [25, Example 1]:

\[
\begin{align*}
g(x) &\rightarrow h(x) \\
c &\rightarrow d \\
b(d) &\rightarrow g(c)
\end{align*}
\]

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

\[
\begin{align*}
G(x) &\rightarrow H(x) \\
H(d) &\rightarrow G(c)
\end{align*}
\]
and unh(R,μ) is:

c → C

Then, for the CS problem τ = (DP(R,μ), R, unh(R,μ), μ♯) we can apply Proc_{NRP}(τ) to remove the second pair in τ by using the following projection⁵:

\[ \pi(G) = 1 \]
\[ \pi(H) = 1 \]

and the following polynomial interpretation:

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

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].

```maude
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

⁵ Since DP_{X}(R,μ) = ∅, we do not need to satisfy (3) in Theorem 12.
Termination Problems Data Base\(^6\) (TPDB): TRS/CSR_Maude/LengthOfFiniteLists_complete.trs. As far as we know, \textsc{mu-term} is the only tool that can prove termination of this system thanks to the CSDP framework presented in this paper\(^7\).

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\(^8\) 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 \textsc{mu-term}. Our tool \textsc{mu-term} participated in the aforementioned CSR category of the 2009 Termination Competition. The results of the competition are summarized in Table 1. Tools APoVE \([12]\) and VMTL \([23]\) implement the 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 APoVE, but \textsc{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 \textsc{mu-term} only.

\(\text{Table 1. 2009 Termination Competition Results (Context-Sensitive Rewriting)}\)

\begin{center}
\begin{tabular}{|l|l|l|}
\hline
Tool & Version & Proved & Average time \\
\hline
APoVE & 34/37 & 3.084s \\
Jambox & 28/37 & 2.292s \\
\textsc{mu-term} & 34/37 & 1.277s \\
VMTL & 29/37 & 6.108s \\
\hline
\end{tabular}
\end{center}

\(^{6}\) http://www.lri.fr/~marche/tpdb/

\(^{7}\) On May 12, 2010, we introduced this system in the online version of APoVE http://aprove.informatik.rwth-aachen.de/, and a timeout occurred after 120 seconds (maximum timeout). \textsc{mu-term} proof can be found in http://zenon.dsic.upv.es/muterm/benchmarks/benchmarks-csr/benchmarks.html

\(^{8}\) 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\textsuperscript{9}, 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 $\text{Proc}_{\text{Coll}}$ 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 $\text{Proc}_{\text{Coll}}$ 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] we have also executed the complete collection of systems of the CSR category\textsuperscript{9}, 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.

\textsuperscript{9} 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 loose their descriptive power), adds a new fresh symbol $U$ which has nothing
to do with the original CS-TRS, and makes the dependency graph harder to
understand.

We have shown a different way to mechanize the context-sensitive dependency
pair approach. The idea is adding a new TRS, the $unhiding$ TRS, which avoids
the extra requirements in [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-
    Cervesato, I., Veith, H., Voronkov, A. (eds.) Proc. of the 15th International Con-
    ference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR’08.
    Arun-Kumar, S., Garg, N. (eds.) Proc. of the 26th Conference on Foundations
    of Software Technology and Theoretical Computer Science, FST&TCS’06. LNCS,
    mation and Computation 208, 922–968 (2010)
    Properties with $mu$-TERM. In: Proc. of the 13th International Conference on Alge-
    braic Methodology and Software Technology, AMAST’10. LNCS, Springer-Verlag
    (2010), to appear
5. Alarcón, B., Lucas, S.: Termination of Innermost Context-Sensitive Rewriting Us-
    ing Dependency Pairs. In: Wolter, F. (ed.) Proc. of the 6th International Sympo-
    Springer-Verlag (2007)
6. Arts, T., Giesl, J.: Termination of Term Rewriting Using Dependency Pairs. The-
    Theoretical Computer Science 360(1), 386–414 (2006)
    Termination of Membership Equational Programs. Higher Order Symbolic Com-
9. Endrullis, J.: Jambox, Automated Termination Proofs For String and Term Rewrit-
    ing (2009), available at http://joerg.endrullis.de/jambox.html
    Terenin, R. (ed.) Proc. of the 20th International Conference on Rewriting Tech-
    (2009)
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

- $u \in \mathcal{M}_{\omega, \mu}$ and
- whenever $u \triangleright_{\mu} s \sqsubseteq_{\mu} t'$ for some terms $s$ and $t'$ with $t' \in \mathcal{M}_{\omega, \mu}$, then $t'$ is an instance of a hidden term and $s = C[t']$ for some hiding context $C[\cdot]$.

Lemma 1 (Hiding Property Lemma [1]). Let $u$ be a term with the hiding property and let $u \not\rightarrow_{\mathcal{R}, \mu} v \sqsubseteq_{\mu} w$ with $w \in \mathcal{M}_{\omega, \mu}$. Then $w$ also has the hiding property.

Proof ([1]). Let $w \triangleright_{\mu} s \sqsubseteq_{\mu} t'$ for some terms $s$ and $t'$ with $t' \in \mathcal{M}_{\omega, \mu}$. Clearly, this also implies $v \triangleright_{\mu} s$. If already $u \triangleright_{\mu} s$, 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 $C[\cdot]$, since $u$ has the hiding property. Otherwise, $u \not\triangleright_{\mu} s$. There must be a rule $t \rightarrow r \in \mathcal{R}$, an active context $D[\cdot]$ (i.e., a context where the hole is at an active position), and a substitution $\delta$ such that $u = D[\delta(t)]$ and $v = D[\delta(r)]$. Clearly, $u \not\triangleright_{\mu} s$ implies $\delta(t) \not\triangleright_{\mu} s$ and $D[\mathcal{C}] \not\triangleright_{\mu} s$. Hence, $u \triangleright_{\mu} s$ means $\delta(r) \triangleright_{\mu} s$. (The root of $s$ cannot be above $\mathcal{C}$ in $D[\mathcal{C}]$ since those positions would be active.) Note that $s$ cannot be at or below a variable position of $r$, because this would imply $\delta(t) \triangleright_{\mu} s$. Thus, $s$ is an instance of a non-variable subterm of $r$ that is at an inactive position. So there is a $r' \not\triangleright_{\mu} x$ 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 $s_{|p} = t_{|p}$. If $p$ is a non-variable position of $r'$, then $\delta(r'_{|p}) = t'_{|p}$ and $r'_{|p}$ is a subterm with defined root at an active position (since $t' \in \mathcal{M}_{\omega, \mu}$ 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 $\mathcal{C}[\mathcal{D}] = r'_{|p}[\mathcal{D}]$ is hiding. So if we define $C[\cdot]$ to be $\delta(\mathcal{C}[\mathcal{D}])$, then $s = \delta(r') = \delta(r')_{|p}[\mathcal{D}] = C[t']$ for the hiding context $C[\cdot]$. On the contrary, if $p$ is not a non-variable position of $r'$, then $p = p_1 p_2$ where $r'_{|p_2}$ 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 $t$, we have $\delta(t) \triangleright \delta(x)$ and thus $u \triangleright \delta(x)$. Due to the minimality of $u$ this implies $u \triangleright \delta(x)$ and $x \in (\text{Var}(\delta(t)) \cap \text{Var}(\delta(x))) \setminus (\text{Var}(\delta(t)) \cup \text{Var}(\delta(x)))$.

Since $u \triangleright \delta(x) \sqsubseteq_{\mu} t'$, the hiding property of $u$ implies that $t'$ is an instance of a hidden term and that $\delta(x) = \mathcal{C}[\mathcal{D}]$ for a hiding context $\mathcal{C}[\cdot]$. Note that since $r'_{|p_1}$ is a variable, the context $\mathcal{C}[\cdot]$ around this variable is also hiding (i.e., $\mathcal{C} = r'_{|p_2}$). Thus, the context $\mathcal{C}[\mathcal{D}] = \delta(\mathcal{C}[\mathcal{D}])$ is hiding as well and $s = \delta(r') = \delta(r')_{|p_2}[\mathcal{D}] = C[t']$. Proposition 1 (Minimal Root Step [2]). Let $\mathcal{R} = (\mathcal{C} \cup \mathcal{D}, \mathcal{R})$ be a TRS and $\mu \in \mathcal{M}_R$. Then for all $t \in \mathcal{M}_{\omega, \mu}$, there exist $t \rightarrow r \in \mathcal{R}$, a substitution $\sigma$ and a term $u \in \mathcal{M}_{\omega, \mu}$ such that $t \triangleright_{\mathcal{R}, \mu, \sigma} u \not\rightarrow_{\mathcal{R}, \mu} u$ and either (1) there is a nonvariable $\mu$-replacing subterm $s$ of $r$ such that $u = \sigma(s)$, or (2) there is $x \in \text{Var}^\mu(r) \setminus \text{Var}^\mu(t)$ such that $\sigma(x) \sqsubseteq_{\mu} u$. 


Theorem 13 (Minimal Sequence in [3]). Let $R$ be a TRS and $\mu \in M_R$. For all $t \in T_{\infty, \mu}$, there is an infinite sequence
\[ t = t_0 \xrightarrow{A}_{R, \mu} \sigma_1(t_1) \xrightarrow{A}_{R, \mu} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{A}_{R, \mu} \sigma_2(t_2) \xrightarrow{A}_{R, \mu} \cdots \]
where, for all $i \geq 1$, $t_i \rightarrow r_i \in R$ are rewrite rules, $\sigma_i$ are substitutions, and terms $t_i \in M_{\infty, \mu}$ are minimal non-$\mu$-terminating terms such that either
1. $t_i = \sigma_i(s_i)$ for some nonvariable term $s_i$ such that $r_i \nsubseteq s_i$, or
2. $\sigma_i(x_i) \supseteq r_i$ for some $x_i \in \text{Var}^\mu(r_i)$ and $t_i = \theta_i(t_i')$ for some $\theta_i \in \text{HT}$ and substitution $\theta_i$.

Theorem 1 (Minimal Sequence). Let $R = (F, R)$ be a TRS and $\mu \in M_R$. For all $t \in T_{\infty, \mu}$, there is an infinite sequence
\[ t = t_0 \xrightarrow{A}_{R, \mu} \sigma_1(t_1) \xrightarrow{A}_{R, \mu} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{A}_{R, \mu} \sigma_2(t_2) \xrightarrow{A}_{R, \mu} \cdots \]
where, for all $i \geq 1$, $t_i \rightarrow r_i \in R$ are rewrite rules, $\sigma_i$ are substitutions, and terms $t_i \in M_{\infty, \mu}$ are minimal non-$\mu$-terminating terms such that either
1. $t_i = \sigma_i(s_i)$ for some nonvariable term $s_i$ such that $r_i \nsubseteq s_i$, or
2. $\sigma_i(x_i) \supseteq r_i$ for some variable $x_i \in \text{Var}^\mu(r_i) \setminus \text{Var}^\mu(t_i)$, $t_i' \in \text{HT}(R, \mu)$, hiding context $C_i[t_i']$, and substitution $\theta_i$.

Proof. Since $T_{\infty, \mu} \subseteq M_{\infty, \mu}$, by Theorem 13 we have a sequence of the form
\[ t = t_0 \xrightarrow{A}_{R, \mu} \sigma_1(t_1) \xrightarrow{A}_{R, \mu} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{A}_{R, \mu} \sigma_2(t_2) \xrightarrow{A}_{R, \mu} \cdots \]
where, for all $i \geq 1$, $t_i \rightarrow r_i \in R$, $\sigma_i$ are substitutions, $t_i \in M_{\infty, \mu}$, and, by Proposition 1, either (1) $t_i = \sigma_i(s_i)$ for some $s_i$ such that $r_i \nsubseteq s_i$, or (2) $\sigma_i(x_i) \supseteq r_i$ for some $x_i \in \text{Var}^\mu(r_i) \setminus \text{Var}^\mu(t_i)$ (and hence $\sigma_i(t_i) \supseteq r_i$ and $\sigma_i(r_i) \supseteq t_i$ as well). If $\sigma_i(x_i) \supseteq r_i$, for some $x_i \in \text{Var}^\mu(r_i) \setminus \text{Var}^\mu(t_i)$, it means that $\sigma_i(t_i) \supseteq C_i[t_i']$.

Since $t \in T_{\infty, \mu}$, it has the hiding property and, by Lemma 1, all $\sigma_i(t_i)$ satisfies the hiding property. Hence, $C_i[t_i] = \theta_i(C_i[t_i'])$ where $\theta_i \in \text{HT}(R, \mu)$ and $C_i[t_i']$ is a hiding context.

B Proof of Theorem 2

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

Proof. Soundness.

By contradiction. If $R$ is not $\mu$-terminating, then there is $t \in T_{\infty, \mu}$. By Theorem 1, there are rules $t_i \rightarrow r_i \in R$, matching substitutions $\sigma_i$, and terms $t_i \in M_{\infty, \mu}$, for $i \geq 1$ such that
\[ t = t_0 \xrightarrow{A}_{R, \mu} \sigma_1(t_1) \xrightarrow{A}_{R, \mu} \sigma_1(r_1) \xrightarrow{\mu} t_1 \xrightarrow{A}_{R, \mu} \sigma_2(t_2) \xrightarrow{A}_{R, \mu} \cdots \]
where either (D1) \( t_i = \sigma_i(s_i) \) for some \( s_i \) such that \( r_i,\mu \vdash t_i \in \mathcal{NHT} \) and hiding context \( C_i[t_i] \). Furthermore, since \( t_{i+1} \in \mathcal{NHT} \) and \( t_i \in \mathcal{NHT} \) (in particular, \( t_0 = t \in \mathcal{NHT} \)), \( \sigma_i(t_i) \in \mathcal{NHT} \) for all \( i \geq 1 \). Note that, since \( t_i \in \mathcal{NHT} \), 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 \( \mathcal{DP}(\mathcal{R}, \mu) \) is a TRS \( \mathcal{P} \) over the signature \( \mathcal{G} = \mathcal{F} \cup \mathcal{D} \) and \( \mu \in \mathcal{M}_{\mathcal{D}, \mathcal{P}} \) as required by Definition 5. Furthermore, \( \mathcal{P}_F = \mathcal{DP}_F(\mathcal{R}, \mu) \) and \( \mathcal{P}_X = \mathcal{DP}_X(\mathcal{R}, \mu) \). We can define an infinite minimal \( \mathcal{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^2 \)-chain using CSDPs \( u_i \rightarrow v_i \) for \( i \geq 1 \), where \( u_i = t_i' \) and

1. \( v_i = s_i' \) if (D1) holds. Since \( t_i \in \mathcal{M}_{\mathcal{D}, \mathcal{P}} \), we have that \( \text{root}(s_i) \in \mathcal{D} \) and, because \( t_i = \sigma_i(s_i) \) and \( \sigma_i(s_i) \rightarrow^* \sigma_{i+1}(t_{i+1}) \), \( \text{ReN}(s_i) \) is \( \mu \)-narrowable [3, Proposition 5]. Furthermore, if we assume that \( s_i \) is a \( \mu \)-replacing subterm of \( t_i \) (i.e., \( t_i \triangleright_\mu s_i \)), then \( \sigma_i(t_i) \triangleright_\mu \sigma_i(s_i) \) which (since \( \sigma_i(s_i) = t_i \in \mathcal{M}_{\mathcal{D}, \mathcal{P}} \)) contradicts that \( \sigma_i(t_i) \in \mathcal{M}_{\mathcal{D}, \mathcal{P}} \). Thus, \( t_i \rightarrow^* v_i \in \mathcal{DP}(\mathcal{R}, \mu) \).

Furthermore, \( t_i' = \sigma_i(v_i) \) is \( \mu \)-terminating. Finally, since \( t_i = \sigma_i(s_i) \rightarrow^* \mathcal{R}, \mu \sigma_{i+1}(t_{i+1}) \) and \( \mu^2 \) extends \( \mu \) to \( \mathcal{F} \cup \mathcal{D} \) by \( \mu^2(f) = \mu(f) \) for all \( f \in \mathcal{D} \), we also have that \( \sigma_i(v_i) = \sigma_i(t_i') \rightarrow^* \mathcal{R}, \mu \sigma_{i+1}(t_{i+1}) \).

2. \( v_i = x_i \) if (D2) holds. Clearly, \( u_i \rightarrow v_i \in \mathcal{DP}_X(\mathcal{R}, \mu) \). As discussed above, \( t_i' \) 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 \( C_i[t_i] \) is an instance of hiding context \( C_i'[t_i] \), then we have that \( \sigma_i(C_i'[t_i]) \rightarrow^* \mathcal{R}, \mu, t_i \). And we also know that \( t_i \) is an instance \( \theta_i(C_i'[t_i]) \) of a hidden term \( t_i' \in \mathcal{NHT}(\mathcal{R}, \mu) \). Thus \( t_i' \rightarrow^* \mathcal{S}_\mu t_i \) and we have \( \theta_i(t_i') \rightarrow^* \mathcal{S}_\mu t_i \). Finally, since \( t_i \rightarrow^* \mathcal{R}, \mu, \sigma_{i+1}(t_{i+1}) \), again we have that \( u_i \rightarrow^* \mathcal{R}, \mu, \sigma_{i+1}(t_{i+1}) \).

Regarding \( \sigma \), w.l.o.g., we can assume that \( \text{Var}(t_i) \cap \text{Var}(t_j) = \emptyset \) 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 \( \mathcal{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^2 \)-chain which contradicts our initial assumption.

Completeness.

By contradiction. If there is an infinite \( \mathcal{DP}(\mathcal{R}, \mu), \mathcal{R}, \text{unh}(\mathcal{R}, \mu), \mu^2 \)-chain, then there is a substitution \( \sigma \) and dependency pairs \( u_i \rightarrow v_i \in \mathcal{DP}(\mathcal{R}, \mu) \) such that

1. \( \sigma(v_i) \rightarrow^* \mathcal{R}, \mu, \sigma(u_{i+1}) \) if \( u_i \rightarrow v_i \in \mathcal{DP}_F(\mathcal{R}, \mu) \), and

2. if \( u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{DP}_X(\mathcal{R}, \mu) \), then \( \sigma(v_i) \rightarrow^* \mathcal{S}_\mu, t_i \rightarrow^* \mathcal{S}_\mu, t_i \rightarrow^* \mathcal{R}, \mu, \sigma(u_{i+1}) \) for \( i \geq 1 \). Now, consider the first dependency pair \( u_1 \rightarrow v_1 \) in the sequence:
1. If \( u_1 \rightarrow v_1 \in \text{DPF}(R, \mu) \), then \( v_1^{\flat} \) is a \( \mu \)-replacing subterm of the right-hand-side \( r_1 \) of a rule \( l_1 \rightarrow r_1 \in R \). Therefore, \( r_1 = C_1[v_1^{\flat}] \) for some \( p_1 \in \mathbb{P}_0 \) and we can perform the \( \mu \)-rewriting step \( t_1 = \sigma(u_1) \rightarrow_{R, \mu} \sigma(r_1) = \sigma(C_1) \sigma(v_1^{\flat}) = s_1 \), where \( \sigma(v_1^{\flat}) = \sigma(v_1^{\flat}) \rightarrow_{R, \mu} \sigma(u_2) \) and \( \sigma(u_2) \) initiates an infinite \( \langle \text{DP}(R, \mu), R, \mathbb{U}(R, \mu), \mu' \rangle \)-chain. Note that \( p_1 \in \mathbb{P}_0^{\mu}(s_1) \).

2. If \( u_1 \rightarrow x \in \text{DPF}(R, \mu) \), then there is a rule \( l_1 \rightarrow r_1 \in R \) such that \( u_1 = \ell_1 \) and \( x \in \text{Var}(r_1) \setminus \text{Var}(\ell_1) \), i.e., \( r_1 = C_1[x] \) for some \( q_1 \in \mathbb{P}_0^{\mu}(r_1) \). Furthermore, if \( \sigma(v_1) = C_1[t_1] \) and \( s_1 \), this means that \( C_1[s] \) is an instance of a hiding context \( C_1[\mathbb{Z}] \). Hence, \( l_1 \rightarrow_{R, \mu} \sigma(u_2) \) and \( \sigma(u_2) \) initiates an infinite \( \langle \text{DP}(R, \mu), R, \mathbb{U}(R, \mu), \mu' \rangle \)-chain. Note that \( p_1 = q_1, p_1' \in \mathbb{P}_0^{\mu}(s_1) \) where \( p_1' \) is the position of the hole in \( C_1[\mathbb{Z}] \).

Since \( \mu(f_1) = \mu(f) \), and \( p_1 \in \mathbb{P}_0^{\mu}(s_1) \), we have that \( s_1 \rightarrow_{R, \mu} t_2 \sigma(u_2)[p_1] = t_2 \) and \( p_1 \in \mathbb{P}_0^{\mu}(t_2) \). Therefore, we can build in this way an infinite \( \mu \)-rewrite sequence
\[
l_1 \rightarrow_{R, \mu} s_1 \rightarrow_{R, \mu} t_2 \rightarrow_{R, \mu} \cdots
\]
which contradicts the \( \mu \)-termination of \( R \).

**C** Proof of Theorem 4

In the following proofs, the notion \( \{ \rightarrow_{-1} \} \) for reduction relations \( \rightarrow_{-1} \) and \( \rightarrow_{-2} \)
means that either \( s \rightarrow_{-1} t \) or \( s \rightarrow_{-2} t \), that is, \( \{ \rightarrow_{-1} \} \cap \{ \rightarrow_{-2} \} = \emptyset \).

**Theorem 4 (Collapsing Pair Transformation).** Let \( \tau = (P, R, \mathcal{S}, \mu) \) be a \( CS \) problem where \( P = (\mathcal{G}, P') \) and \( P_0 \) be given by the following rules:

- \( u \rightarrow U(x) \) for every \( u \rightarrow x \in P_X \),
- \( U(s) \rightarrow U(t) \) for every \( s \rightarrow t \in \mathcal{S}_\mu \), and
- \( U(s) \rightarrow t \) for every \( s \rightarrow t \in \mathcal{S}_\mu \).

Here, \( U \) is a new fresh symbol. Let \( P'' = (\mathcal{G} \cup \{ U \}, P') \) where \( P' = (P \setminus P_0) \cup P_0 \), and \( \mu' \) extends \( \mu \) by \( \mu(U) = \emptyset \). The processor \( \text{Proc}_{\mu}^{\mathcal{S}, \mu} \) given by \( \text{Proc}_{\mu}^{\mathcal{S}, \mu}(\tau) = (P'', R, \mathcal{S}, \mu') \) is sound and complete.

**Proof.** Soundness. We prove first that the existence of an infinite minimal \( (P, R, \mathcal{S}, \mu) \)-chain implies the existence of an infinite minimal \( (P', R, \mathcal{S}, \mu') \)-chain.

First, note that \( P'' \) is well-defined as a TRS. Consider an infinite minimal \( (P, R, \mathcal{S}, \mu) \)-chain \( \alpha \):
\[
\sigma(u_1) \quad \sigma(v_1^{\flat}) \rightarrow_{-1} \sigma(u_2) \quad \cdots
\]

...
for some substitution \( \sigma \), where, for all \( i \geq 1 \), \( t_i \) is \( \mu \)-terminating and, (1) if \( u_i \rightarrow v_i \in \mathcal{P}_0 \), then \( t_i = \sigma(v_i) \) and the chain is maintained unchanged (2) if \( u_i \rightarrow v_i = u_i \rightarrow x_i \in \mathcal{P}_X \), then \( \sigma(v_i) \leftarrow_{\mathcal{S}_{x_i, \mu}} s_i \leftarrow_{\mathcal{S}_{x_i, \mu}} t_i \). Note that, instead of \( u_i \rightarrow x_i \), we have \( u_i \rightarrow U(x_i) \); for all rules \( s \rightarrow t \) such that \( s \vdash_{\mu} t (s \rightarrow t \in \mathcal{S}_{\mathcal{P}}) \) we have \( U(s) \rightarrow U(t) \in \mathcal{P}' \); and for all rules \( s \rightarrow t \in \mathcal{S}_f \) we have \( U(s) \rightarrow t \in \mathcal{P}' \). Therefore, for all terms \( s, t \), \( s \vdash_{\mu} U(t) \) (where we only use the rules in \( \mathcal{P}' \) which are of the form \( U(s) \rightarrow U(t) \)), and \( s \vdash_{\mu} U(t) \) if and only if \( U(s) \rightarrow_{\mu} t \). Hence, we obtain:

\[
\begin{align*}
A & \xrightarrow{\mathcal{P}_0, \mu} U(\sigma(x_i)) \text{ a collapsing pair is applied} \\
A & \xrightarrow{\mathcal{P}_0, \mu} U(s_i) \quad \text{we apply } \mathcal{S}_{x_i, \mu} \text{-rules} \\
A & \xrightarrow{\mathcal{P}_0, \mu} t_i \quad \text{we apply a } \mathcal{S}_{f} \text{-rule}
\end{align*}
\]

Thus, we obtain an infinite minimal \((\mathcal{P}', \mathcal{R}, \mathcal{S}, \mu')\)-chain, as desired. In particular, we note that all steps with \( \mathcal{P}_0 \) are performed at the root, we do not require any reduction below symbol \( U \), hence \( \mu'(U) = \emptyset \) 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 \( u_i \rightarrow v_i \in \mathcal{P}' \setminus \mathcal{P}_0 \) and \( \sigma(v_i) \leftarrow_{\mathcal{R}_\mu} \sigma(u_{i+1}) \), then \( u_i \rightarrow v_i \in \mathcal{P} \)
2. If \( u_i \rightarrow v_i = u_i \rightarrow U(x_i) \in \mathcal{P}' \) and \( \sigma(u_i) \leftarrow_{\mathcal{R}_\mu} \sigma(v_{i+1}) \) where \( \sigma(v_{i+1}) = U(u_{i+1}) \), then there is a pair \( u_i \rightarrow x_i \in \mathcal{P} \) such that \( \sigma(x_i) = v'_{i+1} \)
3. If \( u_i \rightarrow v_i = U(s) \rightarrow U(t) \in \mathcal{P}' \) then there is a pair in \( \mathcal{S}_{x_i, \mu} \); \( s \rightarrow t \in \mathcal{S}_{\mathcal{P}} \), and
4. If \( u_i \rightarrow v_i = U(s) \rightarrow t \in \mathcal{P}' \) then there is a pair \( s \rightarrow t \in \mathcal{S}_f \).

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 \( \mathcal{R} = (\mathcal{C} \sqcup D, \mathcal{R}) \) and \( \mathcal{S} = (\mathcal{H}, \mathcal{S}) \). Assume that (1) all the rules in \( \mathcal{S}_f \) are noncollapsing, i.e., for all \( s \rightarrow t \in \mathcal{S}_f \), \( t \notin \mathcal{X} \) (2) \( \{\text{root}(t) \mid s \rightarrow t \in \mathcal{S}_f \} \sqcup D = \emptyset \) and (3) for all \( s \rightarrow t \in \mathcal{S}_0 \), we have that \( s = \{(s_1, \ldots, s_k) \text{ and } t = (x_1, \ldots, x_j) \text{ for some } k \in \mathcal{N}, \text{ function symbols } f, g \in \mathcal{H}, \text{ and terms } s_1, \ldots, s_k \). Then, the processors \( \text{Proc}_{\mathcal{CS}}(\tau) \) given by

\[
\text{Proc}_{\mathcal{CS}}(\tau) = \begin{cases} \\
\emptyset & \text{if } \mathcal{P} = \mathcal{P}_f \\
\{(P, \mathcal{R}, \mathcal{S}, \mu)\} & \text{otherwise}
\end{cases}
\]

is sound and complete.
Proof. If \( P = P^1 \), then we proceed by contradiction. Assume that there is an infinite chain which only uses dependency pairs \( u_i \rightarrow x_i \in P^1 \) for all \( i \geq 1 \). Let \( t_i = \text{root}(u_i) \) for \( i \geq 1 \). Then, by definition of \( P^1 \), for all \( i \geq 1 \) there is \( j_i \in \{1, \ldots, \sigma(t_i)\} \setminus \mu(t_i) \) such that \( u_i |_{j_i} \geq x_i \). We have that \( \sigma(u_i) |_{j_i} \geq \sigma(x_i) \rightarrow^{s_{j_i}} \sigma(u_{i+1}) \), \( s_{i} \rightarrow^{s_{j_i}} t_{i} \) for some terms \( s_{i}, t_{i} \) such that \( t_{i} \rightarrow^{s_{j_i}} \sigma(u_{i+1}) \). Since \( s_{i} = \theta(s') \) and \( t_{i} = \theta(t') \) for some \( s' \rightarrow t' \in S_{f} \) and substitution \( \theta \), by (1), we have that \( \text{root}(t_{i}) = \text{root}(t'_{i}) \). By (2), we have that \( \text{root}(t_{i}) \notin D \), i.e., \( t_{i} \) cannot be \( \mu \)-rewritten at the root by using the rules in \( R \). Hence, \( \text{root}(t_{i}) = \text{root}(u_{i+1}) = t_{i+1} \) and \( j_{i+1} \notin \mu(t_{i+1}) \). Note that \( \sigma(x_i) \geq s_i \). Since \( t_{i} \rightarrow^{s_{j_i}} \sigma(u_{i+1}) \) and no \( \mu \)-rewriting is possible on the \( j_{i+1} \)-th immediate subtree \( t_{i+1} |_{j_{i+1}} \) of \( t_{i} \) and also \( s_{i} \) and \( t_{i+1} \) only differ in the root symbol (due to (3)), it follows that \( \sigma(u_{i+1}) |_{j_{i+1}} \geq s_{i} \), \( s_{i} |_{j_{i+1}} = t_{i+1} |_{j_{i+1}} \) (due to (3)) and \( t_{i+1} |_{j_{i+1}} = \sigma(u_{i+1}) |_{j_{i+1}} \geq \sigma(x_{i+1}) \), i.e., \( \sigma(x_{i}) \triangleright \sigma(x_{i+1}) \) for all \( i \geq 1 \). We get an infinite sequence \( \sigma(x_{1}) \triangleright \sigma(x_{2}) \triangleright \cdots \) which contradicts well-foundedness of \( \triangleright \).

E Proof of Theorem 6

Theorem 6 (SCC Processor). Let \( \tau = (P, R, S, \mu) \) be a CS problem. Then, the processor \( \text{Proc}_{SCC} \) given by

\[
\text{Proc}_{SCC}(P, R, S, \mu) = \{ (Q, R, S_{Q}, \mu) \mid Q \text{ are the pairs of an SCC in } G(P, R, S, \mu) \}
\]

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

Proof. We prove soundness by contradiction. Assume that \( \text{Proc}_{SCC} \) is not sound. Then, there is a CS problem \( \tau = (P, R, S, \mu) \) such that, for all \( \tau' \in \text{Proc}_{SCC}(\tau) \), \( \tau' \) is finite but \( \tau \) is not finite. Thus, there is an infinite minimal \((P', R', S', \mu)-\text{chain} A \). Since \( P \) contains a finite number of pairs, there is \( P' \subseteq P \) and a tail \( B \) of \( A \), which is an infinite minimal \((P', R', S', \mu)-\text{chain} \) where all pairs in \( P' \) are infinitely often used. According to Definition 7, this means that \( P' \) is a cycle in \( G(P, R, S, \mu) \). Hence \( P' \) belongs to some SCC with nodes in \( Q \), i.e., \( P' \subseteq Q \). Thus, \( B \) is an infinite minimal \((Q, R, S, \mu)-\text{chain} \) -chain, i.e., \( \tau' = (Q, R, S, \mu) \) is not finite. Since \( \tau' \in \text{Proc}_{SCC}(\tau) \), we obtain a contradiction.

With regard to completeness, since \( Q \subseteq P \) for some SCC in \( G(P, R, S, \mu) \) with nodes in \( Q \), every infinite minimal \((Q, R, S, \mu)-\text{chain} \) is an infinite minimal \((P, R, S, \mu)-\text{chain} \). Hence, the processor is complete as well.

F Proof of Theorem 7

We need some previous results to prove Theorem 7.

Proposition 2 (\( \text{TCAP}_{\mu}^{R} \text{ Property} \) [3]). Let \( R = (F, R) \) be a TRS, \( G \) be a signature and \( \mu \in M_{R, G} \). Let \( t, u \in T(F \cup G, X) \) be such that \( \text{Var}(t) \cap \text{Var}(u) = \emptyset \). If there are substitutions \( \theta \) and \( \theta' \) such that \( \theta(t) \rightarrow^{R, \mu} \theta'(u) \), then \( \text{TCAP}_{\mu}^{R}(t) \) and \( u \) unify.
Theorem 7 (Approximation of the CS Graph). Let \( R, P \) and \( S \) be TRSs and \( \mu \in M_{R,P,S} \). The estimated CS graph \( EG(P,R,S,\mu) \) contains the CS graph \( G(P,R,S,\mu) \).

Proof. Direct as a consequence of Definition 5, Definition 7 and Proposition 2.

G Proof of Theorem 8

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

\[
Proc_{SCC}(\tau) = \{(Q,R,S,\mu) | Q \text{ contains the pairs of an SCC in } EG(P,R,S,\mu)\}
\]

where

- \( S_Q = \emptyset \) if \( Q_X = \emptyset \).
- \( S_Q = S_Q \cup \{s \to t | s \to t \in S_Q, \text{TCap}_R(t) \text{ and } u' \text{ unify for some } u' \to v' \in Q \text{ if } Q_X \neq \emptyset \} \)

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

1. \( P \subseteq \geq \cup \sqsupset \), \( R \subseteq \geq \), and
2. whenever \( P_X \neq \emptyset \) we have that \( S \subseteq \geq \cup \sqsupset \cup \succeq \).

Let \( P_{\geq} = \{u \to v \in P | u \sqsupset v\} \) and \( S_{\geq} = \{s \to t \in S | s \succeq t\} \). Then, the processor \( Proc_{RT} \) given by

\[
Proc_{RT}(\tau) = \begin{cases} 
\{(P \setminus P_{\geq}, R, S \setminus S_{\geq}, \mu) \} & \text{if (1) and (2) hold} \\
\{(P, R, S, \mu)\} & \text{otherwise}
\end{cases}
\]

is sound and complete.

Proof. Completeness is obvious, since \( P \setminus P_{\geq} \subseteq P \). Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal \( (P,R,S,\mu) \)-chain \( A \), but there is no infinite minimal \( (P \setminus P_{\geq}, S \setminus S_{\geq}, R, \mu) \)-chain. Due to the finiteness of \( P \) and \( S \), we can assume that there are subsets \( Q \subseteq P \) and \( T \subseteq S \) such that \( A \) has a tail \( B \)

\[
\sigma(a_1) \xrightarrow{\text{unify}} \sigma(a_2) \xrightarrow{\text{unify}} \sigma(a_3) \xrightarrow{\text{unify}} \ldots
\]
for some substitution σ, where all pairs in Q and all rules in T are infinitely often used (note that, if T ≠ ∅, then Tq ≠ ∅ and Qx ≠ ∅), and, for all i ≥ 1, (1) if ui → vi ∈ Q, then ti = σ(vi) and (2) if ui = vi = xi ∈ Qx, then σ(ui) → Tq o Aσ Rσ(µ) o Aσ Rσ ti.

Since u.(i ≥ 1 ∨) vi for all ui → vi ∈ Q ≤ P, by stability of ≥ and ⊢, we have σ(µ) (i ≥ 1 ∨) σ(vi) for all i ≥ 1.

No pair u → v ∈ Q satisfies that u ⊢ v and no rule ℓ → r ∈ T satisfies s ⊢ t. Otherwise, we get a contradiction by considering the following two cases:

1. If ui → vi ∈ Q, then ti = σ(vi) → Sσ(µ) σ(ui+1) and ti ≥ σ(ui+1). Since we have σ(µ)(i ≥ 1 ∨) σ(vi) = ti, by using transitivity of ≥ and compatibility between ≥ and ⊢, we conclude that σ(µ)(i ≥ 1 ∨) σ(ui+1).

2. If ui = vi = xi ∈ Q (which is not empty whenever T is not empty), then σ(vi) = σ(xi) → Aσ Rσ(µ) σ(x1) Since ti (i ≥ 1 ∨) ≥ ≥ ≥ i for all ti → ri ∈ Tq, we have σ(vi) = σ(xi)(i ≥ 1 ∨) ≥ ≥ ≥ 1. Furthermore, we are assuming that ℓ (i ≥ 1 ∨) ≥ ≥ ≥ i for all ℓ → r ∈ T. Since σ(µ)(i ≥ 1 ∨) σ(xj) for some xj → tj ∈ Tq and, by stability of the quasi-orderings we have that σ(xj) (i ≥ 1 ∨) ≥ ≥ ≥ 1. Hence, by transitivity of ≥ (and compatibility of ≥, ⊢ and ≥), we have σ(vi) = σ(xi)(i ≥ 1 ∨) ≥ ≥ ≥ 1. Since σ(µ)(i ≥ 1 ∨) σ(xj), we also have that, for all i ≥ 1, σ(ti) ≥ σ(xj). Therefore, again by transitivity of ≥ and compatibility of ≥, ≥ and ⊢, we conclude that σ(µ)(i ≥ 1 ∨) (i ≥ 1 ∨) ≥ ≥ ≥ 1 and hence σ(µ)(i ≥ 1 ∨) ≥ ≥ ≥ 1.

Since u → v and ℓ → r occur infinitely often in B, there is an infinite set I ⊆ N of pairs such that σ(µ)(i ∈ I) ≥ ≥ ≥ 1 for all i ∈ I. Thus, by using the compatibility conditions of the µ-reduction triple, we obtain an infinite decreasing ⊢-sequence which contradicts well-foundedness of ⊢.

Therefore, B is an infinite minimal (P ∖ Pq, R, S ∖ Sq, µ)-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 µ-Interpretation [16]). Let (F, R) be a TRS, µ ∈ M and A ⊆ F. Let > be an arbitrary total ordering over T(F ∪ {⊥, c}, X). Then ⊥ is a fresh constant symbol and c is a fresh binary symbol. The basic µ-interpretation I0,Δ,µ is a mapping from µ-terminating terms in T(F, X) to terms in T(F ∪ {⊥, c}, X) defined as follows:

\[
I_{0,\Delta,\mu}(t) = \begin{cases} 
  t & \text{if } t \in X \\
  f(I_{0,\Delta,\mu}(t_1)), \ldots, I_{0,\Delta,\mu}(t_k) & \text{if } t = f(t_1, \ldots, t_k) \\
  \mathcal{c}(I_{0,\Delta,\mu}(t_1)), \ldots, I_{0,\Delta,\mu}(t_k), t' & \text{if } t = \mathcal{c}(t_1, \ldots, t_k) \\
  I_{0,\Delta,\mu}(t) & \text{if } t \in \Delta
\end{cases}
\]
where $I_{0,\Delta,\mu}(t) = \begin{cases} I_{0,\Delta,\mu}(t) & \text{if } i \in \mu(t) \\ t & \text{if } i \notin \mu(t) \end{cases}$

$\sigma = \text{order } ((I_{0,\Delta,\mu}(u) \mid t \rightarrow_{\mu} u))$

$\text{order}(T) = \begin{cases} \bot, & \text{if } T = \emptyset \\ \psi(t,\text{order}(T \setminus \{t\})) & \text{if } t \text{ is minimal in } T \text{ w.r.t. } \succ \end{cases}$

Lemma 2. [16] Let $R = (F, R)$ be a TRS, $\mu \in MF$ and $t$ in $T(F, X)$. If $t$ is $\mu$-terminating then $I_{0,\Delta,\mu}$ is well-defined.

Proof. According to Definition 18, to obtain an infinite term as result of $I_{0,\Delta,\mu}(t)$, we would have to perform an infinite number of applications of the function $I_{0,\Delta,\mu}$, which is not possible. Hence, this is not possible.

Definition 19. [16] Let $R = (F, R)$ be a TRS, $\mu \in MF$ and $\sigma$ be a substitution. Let $\Delta \subseteq F$. We denote by $\sigma_{I_{0,\Delta,\mu}} : T(F, X) \to T(F, X)$ a function that, given a term $t$ replaces occurrences of $x \in \text{Var}(t)$ at position $p$ in $t$ by either $I_{0,\Delta,\mu}((\sigma(x)))$ if $p \in \text{Pos}(t)$ and $\sigma(x)$ is $\mu$-terminating, or $\sigma(x)$ otherwise.

Proposition 3. [16] Let $R = (F, R)$ be a TRS, $\mu \in MF$ and $\sigma$ be a substitution. Let $\Delta \subseteq F$. Let $t$ be a term such that $\text{Var}(t) \cap \text{Var}(\sigma(t)) = \emptyset$. Let $\sigma_{I_{0,\Delta,\mu}}$ be a substitution given by

$$\sigma_{I_{0,\Delta,\mu}}(x) = \begin{cases} I_{0,\Delta,\mu}(\sigma(x)) & \text{if } x \in \text{Var}(t) \text{ and } \sigma(x) \text{ is } \mu\text{-terminating} \\ \sigma(x) & \text{otherwise} \end{cases}$$

Then, $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(t)$.

Proof. By structural induction on $t$.

- If $t = x \in X$ we have two possibilities: (1) if $\sigma(x)$ is $\mu$-terminating, then $\sigma_{I_{0,\Delta,\mu}} = I_{0,\Delta,\mu}(\sigma(x))$ and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(\sigma(x))$ (since $x \in \text{Var}(t)$ and $\sigma(x)$ is $\mu$-terminating), and (2) if $\sigma(x)$ is non-$\mu$-terminating, then $\sigma_{I_{0,\Delta,\mu}} = \sigma(x)$ and $\sigma_{I_{0,\Delta,\mu}}(t) = \sigma(x)$ (since $x \in \text{Var}(t)$ and $\sigma(x)$ is non-$\mu$-terminating)

- If $t = f(t_1, \ldots, t_k)$ we have two possibilities: (1) if $i \in \mu(t)$ we can apply the induction hypothesis to $t_i$ and obtain that $\sigma_{I_{0,\Delta,\mu}}(t_i) = I_{0,\Delta,\mu}(t_i)$, and (2) if $i \notin \mu(t)$ and $i \notin \mu(t)$ then $\sigma_{I_{0,\Delta,\mu}}(t_i) = I_{0,\Delta,\mu}(t_i) = \sigma(t_i)$.

Since $\text{Var}(t) \cap \text{Var}(\sigma(t)) = \emptyset$ we can ensure that for all variable $x$ in $t$: (1) if $x \in \text{Var}(t)$ and $\sigma(x)$ is $\mu$-terminating then $x \notin \text{Var}(\sigma(t))$ and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(\sigma(x))$, and (2) if $x \notin \text{Var}(t)$ (then $x \notin \text{Var}(\sigma(t))$) or $\sigma(x)$ is non-$\mu$-terminating then and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(t) = \sigma(t)$.

Lemma 3. [16] Let $R = (F, R)$ be a TRS, $\mu \in MF$ and $\Delta \subseteq F$. Let $t$ be a term and $\sigma$ be a substitution. If $\sigma(t)$ is $\mu$-terminating, then $I_{0,\Delta,\mu}(\sigma(t)) = I_{0,\Delta,\mu}(\sigma(t))$.

Proof. By structural induction on $t$.

- If $t = x \in X$ we have two possibilities: (1) if $\sigma(x)$ is $\mu$-terminating, then $\sigma_{I_{0,\Delta,\mu}} = I_{0,\Delta,\mu}(\sigma(x))$ and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(\sigma(x))$ (since $x \in \text{Var}(t)$ and $\sigma(x)$ is $\mu$-terminating), and (2) if $\sigma(x)$ is non-$\mu$-terminating, then $\sigma_{I_{0,\Delta,\mu}} = \sigma(x)$ and $\sigma_{I_{0,\Delta,\mu}}(t) = \sigma(x)$ (since $x \in \text{Var}(t)$ and $\sigma(x)$ is non-$\mu$-terminating)

- If $t = f(t_1, \ldots, t_k)$ we have two possibilities: (1) if $i \in \mu(t)$ we can apply the induction hypothesis to $t_i$ and obtain that $\sigma_{I_{0,\Delta,\mu}}(t_i) = I_{0,\Delta,\mu}(t_i)$, and (2) if $i \notin \mu(t)$ and $i \notin \mu(t)$ then $\sigma_{I_{0,\Delta,\mu}}(t_i) = I_{0,\Delta,\mu}(t_i) = \sigma(t_i)$.

Since $\text{Var}(t) \cap \text{Var}(\sigma(t)) = \emptyset$ we can ensure that for all variable $x$ in $t$: (1) if $x \in \text{Var}(t)$ and $\sigma(x)$ is $\mu$-terminating then $x \notin \text{Var}(\sigma(t))$ and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(\sigma(x))$, and (2) if $x \notin \text{Var}(t)$ (then $x \notin \text{Var}(\sigma(t))$) or $\sigma(x)$ is non-$\mu$-terminating then and $\sigma_{I_{0,\Delta,\mu}}(t) = I_{0,\Delta,\mu}(t) = \sigma(t)$.
Proof. By structural induction on $t$:

- If $t$ is a variable then $I_{0,\Delta,\mu}(\sigma(t)) = \sigma|_{t_{0},\Delta,\mu,\nu}(t)$.
- If $t = (t_1,\ldots,t_k)$ then
  
  - If $t \in \Delta$ then $I_{0,\Delta,\mu}(\sigma(t)) = f(I_{0,\Delta,\mu,\nu}(\sigma(t_1)),\ldots,I_{0,\Delta,\mu,\nu}(\sigma(t_k)))$. Terms $\sigma(t_i)$ are $\mu$-terminating for $i \in \mu(t)$. By induction hypothesis, for all terms $t_i$, s.t. $i \in \mu(t)$, we have $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) = I_{0,\Delta,\mu}(\sigma(t_i))$ (by the induction hypothesis).
  
  - If $t \notin \Delta$ then $I_{0,\Delta,\mu}(\sigma(t)) = G \cup F \cup H \rightarrow \sigma(t)$.

Terms $\sigma(t_i)$ are $\mu$-terminating for $i \in \mu(t)$. By induction hypothesis, for all terms $t_i$, s.t. $i \in \mu(t)$, we have $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) = I_{0,\Delta,\mu}(\sigma(t_i))$ (by the induction hypothesis).

This implies $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) \rightarrow_{\mu(t)} I_{0,\Delta,\mu,\nu}(r(t))$. 

- If $t \notin \Delta$ then $I_{0,\Delta,\mu}(\sigma(t)) = c(l(I_{0,\Delta,\mu,\nu}(\sigma(t_1)),\ldots,I_{0,\Delta,\mu,\nu}(\sigma(t_k))))$, for some $r(t)$. Applying a $C_0$ step to this term, we obtain again the term $l(I_{0,\Delta,\mu,\nu}(\sigma(t_1)),\ldots,I_{0,\Delta,\mu,\nu}(\sigma(t_k)))$, and using the previous item result, we get $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) \rightarrow_{\mu(t)} I_{0,\Delta,\mu,\nu}(r(t))$.

Then we conclude $I_{0,\Delta,\mu}(\sigma(t)) \rightarrow_{\mu(t)} I_{0,\Delta,\mu,\nu}(t)$.

The second part of the lemma is proved similarly. If $t$ is a variable then $I_{0,\Delta,\mu}(\sigma(t)) = \sigma|_{t_{0},\Delta,\nu}(t)$. Now let $t = (t_1,\ldots,t_k)$. Since $t \in \Delta$, $I_{0,\Delta,\mu}(\sigma(t)) = I_{0,\Delta,\mu}(\sigma(t_1),\ldots,\sigma(t_k)) = f(I_{0,\Delta,\mu,\nu}(\sigma(t_1)),\ldots,I_{0,\Delta,\mu,\nu}(\sigma(t_k)))$. For $i \in \mu(t)$, we have $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) = I_{0,\Delta,\mu}(\sigma(t_i)) = \sigma|_{t_{0},\Delta,\nu}(t_i)$ by the induction hypothesis. For $i \notin \mu(t)$, we have $I_{0,\Delta,\mu,\nu}(\sigma(t_i)) = \sigma(t_i)$. This implies that $l(I_{0,\Delta,\mu,\nu}(\sigma(t_1)),\ldots,I_{0,\Delta,\mu,\nu}(\sigma(t_k))) = \sigma(t)$.

Proposition 4. [16] Let $R = (\mathcal{F},R)$ be a CRS and $\mu \in M_R$. Let $\Delta \subseteq \mathcal{F}$. For all $\mu$-terminating terms $t \in T(F,\Delta)$, we have $I_{0,\Delta,\mu}(t) \rightarrow_{\mu(t)} t$.

Proof. By structural induction on $t$:

- If $t = x \in X$ then $I_{0,\Delta,\mu}(x) = x$.
- If $t = (t_1,\ldots,t_k)$, we have that $I_{0,\Delta,\mu}(t_i) \rightarrow_{\mu(t)} t_i$ if $i \in \mu(t)$ (by the induction hypothesis), and if $i \notin \mu(t)$, $t_i$ is maintained inalterable. We have two possibilities:
  
  - If $t \in \Delta$ then $I_{0,\Delta,\mu}(f(t_1,\ldots,t_k)) = f(t'_1,\ldots,t'_k)$ where by the induction hypothesis we have that $t'_i \rightarrow_{\mu(t)} t_i$. Hence, $f(t'_1,\ldots,t'_k) \rightarrow_{\mu(t)} f(t_1,\ldots,t_k)$.
  
  - If $t \notin \Delta$ then $I_{0,\Delta,\mu}(f(t_1,\ldots,t_k)) = c(l(t'_1,\ldots,t'_k))$, and applying only $C_0$ steps we have $c(l(t'_1,\ldots,t'_k)) \rightarrow_{\mu(t)} f(t_1,\ldots,t_k)$.

Lemma 4. Let $\sigma = (P,R,S,\mu)$ be a CS problem where $P = (G,P), R = (F,R)$ and $S = (H,S)$. Let $P \cup \{P \mid P \in P \cup R \subseteq P \mid \nu(S) \}$ be strongly conservative, and $\Delta = (G \cup F \cup H) \setminus \{root(t) \mid \epsilon \rightarrow t \in R \cup \{P \mid P \in P \cup R \subseteq P \mid \nu(S) \})$, and $s$ s.t. $s \rightarrow_{\mu,s} t$ then $I_{0,\Delta,\mu}(s) \rightarrow_{\mu,s} \mu \rightarrow_{\Delta} I_{0,\Delta,\mu}(t)$.

Proof. By induction on the position $p$ of the redex in $s \rightarrow_{\mu,s} t$. First assume that $\sigma(t) \notin \Delta$ and $p = A$ (and therefore $\epsilon \rightarrow t \in U^{\mu}(P \cup R \cup S,\mu)$). So we have $s = \sigma(t) \rightarrow_{\mu,s} \sigma(r(t)) = t$ for some substitution $\sigma$. Moreover, for all subterms $\tau'$ at $\mu$-replacing positions of $r$, $\text{root}(\tau') \notin \Delta$ by definition of $\Delta$. Since $\text{Var}(t) \cap \text{Var}(r(t)) = \emptyset$ and $\text{Var}(t') \cap \text{Var}(r(t)) = \emptyset$, we have by Proposition 3 that $\sigma|_{t_{0},\Delta,\mu}(t) \rightarrow_{\mu,s} \sigma(r(t))$ and $\sigma|_{t_{0},\Delta,\mu}(\tau') \rightarrow_{\mu,s} \sigma(r(t'))$. Moreover, for all
variables \( x \) we have \( \sigma^{I_\mu}_X(x) \sim_{C} \sigma^{I_\mu}_X(x) \). To see this, note that by strong conservativity, \( \sigma^{I_\mu}_X \) and \( \sigma^{I_\mu}_X \) only differ on variables \( x \in \Var^\mu(r) \setminus \Var^\mu(r) \). Here, we have \( \sigma^{I_\mu}_X(x) = I_{0, \mu}(\sigma(x)) \sim_{C} \sigma^{I_\mu}_X(x) = \sigma(x) = \sigma^{I_\mu}_X(x) \) by Lemma 3. Hence,

\[
\begin{align*}
I_{0, \mu}(t) &= I_{0, \mu}(\sigma(t)) \\
&\sim_{C} \sigma^{I_\mu}_X(t) \\
&= \sigma^{I_\mu}_X(t) \\
&= I_{0, \mu}(t)
\end{align*}
\]

Now let the case where \( \text{root}(s) \in \Delta \) and \( p \neq A \). Hence, \( s = (s_1, \ldots, s_n) \), \( t = (f(s_1, \ldots, s_n), i \in \mu(f)) \), and \( s_1 \sim_{\{\ell_{i-1}\}_\mu} t_i \). The induction hypothesis implies \( I_{0, \mu}(s) \sim_{\{\ell_{i-1}\}_\mu} I_{0, \mu}(t_i) \) and hence, \( I_{0, \mu}(s) \sim_{\{\ell_{i-1}\}_\mu} I_{0, \mu}(t_i) \). Finally, we consider the case \( \text{root}(s) \notin \Delta \). In this case, \( I_{0, \mu}(t) \in \text{order } (I_{0, \mu}(u) \mid s \sim_{\Gamma, \mu} u) \) because \( s \sim_{\Gamma, \mu} t \). By applying \( C_r \) rules, we get \( I_{0, \mu}(s) \sim_{C_r} I_{0, \mu}(t) \).

**Definition 20** \( (\mu\text{-}\text{Interpretation}) [16] \). Let \( R = (F, R) \) be a TRS, \( \mu \in M_F \) and \( \Delta \subseteq F \). Let \( \succ \) be an arbitrary total ordering over \( T(F \cup \{\bot, \epsilon\}, \mathcal{X}) \) where \( \bot \) is a fresh constant symbol and \( \epsilon \) is a fresh binary symbol (with \( \mu(\epsilon) = \{1, 2\} \)). The \( \mu\text{-}\text{Interpretation } I_{\mu} \) is a mapping from arbitrary terms in \( T(F, \mathcal{X}) \) to terms in \( T(F \cup \{\bot, \epsilon\}, \mathcal{X}) \) defined as follows:

\[
I_{\mu}(t) = \begin{cases} 
    t & \text{if } t \in \mathcal{X} \\
    (I_{\mu}(t_1), \ldots, I_{\mu}(t_k)) & \text{if } t = (t_1, \ldots, t_k) \text{ and } t \notin \Delta \text{ or } t \text{ is non-}\mu\text{-}\text{terminating} \\
    \ell(t_1, \ldots, t_k) & \text{if } t = (t_1, \ldots, t_k) \text{ and } t \notin \Delta \text{ and } t \text{ is } \mu\text{-}\text{terminating}
\end{cases}
\]

where \( \ell = \text{order } ((I_{\mu}(u) \mid t \sim_{\Gamma, \mu} u)) \) and \( \ell(T) \) is minimal in \( T \) w.r.t. \( \succ \).

**Lemma 5.** [16] Let \( R = (F, R) \) be a TRS, \( \mu \in M_F \), \( \Gamma = \bigcup_{\mu \neq 1} \text{Fun}^\mu(r) \), and \( \Delta \subseteq F \). If \( \Gamma \subseteq \Delta \) then \( I_{\mu} \) is well-defined.

**Proof.** According to Definition 20, to obtain an infinite term as the result of \( I_{\mu}(t) \) for a given term \( t \), we would have to perform an infinite number of applications of \( \text{order } ((I_{\mu}(u) \mid t \sim_{\Gamma, \mu} u)) \). This means that \( t \) is \( \mu\text{-}\text{terminating} \) and that there exists an infinite sequence (extracted from Definition 20) of the form \( t = u_1 \in t_1 \sim_{\Gamma, \mu} u_2 \in t_2 \sim_{\Gamma, \mu} u_3 \ldots \) where \( \text{root}(t) \notin \Delta \) and \( t_i \) is \( \mu\text{-}\text{terminating} \) for all \( i \geq 1 \). Without loss of generality we can assume that \( t_i \neq t_{i+1} \) (otherwise, we simply consider the modified sequence \( \cdots \in t_{i-1} \sim_{\Gamma, \mu} u_i \in t_{i+1} \sim_{\Gamma, \mu} u_{i+1} \cdots \)). For \( i \geq 1 \), there is a rule \( \ell \rightarrow r \), an active position \( p \)
of $t_i$, and a substitution $\sigma$ such that $t_i = C[\sigma(t)]_{\mu} \hookrightarrow R, \mu C[\sigma(r)]_{\mu} = u_{i+1} \triangleright t_{i+1}$.

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 $s \notin X$ and $r \triangleright s$ such that $\sigma(s) = t_{i+1}$. Since $\text{root}(s) \notin \Delta$ and $F \cup \mu(r) \subseteq \Delta$, we have $r \triangleright \mu s$ and hence, $u_{i+1} \triangleright \mu t_{i+1}$.

The resulting sequence is: $t = u_1 \triangleright \mu t_1 \hookrightarrow R, \mu u_2 \triangleright \mu t_2 \hookrightarrow R, \mu u_3 \ldots$, contradicting the termination of $t$.

**Definition 21.** [16] Let $R = (F, R)$ be a TRS, $\mu \in M_F$ and $\sigma$ be a substitution.

Let $\Delta \subseteq F$. We denote by $\sigma_{\Delta, \mu}$ a substitution such that, given a term $t$ replaces occurrences of $x$ by $I_{\Delta, \mu}(\sigma(x))$.

**Lemma 6.** [16] Let $R = (F, R)$ be a TRS and $\mu \in M_F$. Let $\Delta \subseteq F$. If all subterms $t'$ of $t$ at non-$\mu$-replacing positions are from $T(\Delta, X)$ then we have $I_{\Delta, \mu}(\sigma(t)) \hookrightarrow^{*} \sigma_{\Delta, \mu}(t)$.

**Proof.** We use the induction on $t$. If $t$ is a variable then $I_{\Delta, \mu}(\sigma(t)) = I_{\Delta, \mu}(\sigma(t))$.

Now, let $t = (t_1, \ldots, t_k)$. By induction hypothesis $I_{\Delta, \mu}(\sigma(t_i)) \hookrightarrow^{*} \sigma_{\Delta, \mu}(t_i)$ for $1 \leq i \leq k$. Moreover, by hypothesis: whenever $i \notin \mu(t)$ then $t_i$ contains only $\Delta$-symbols. Then $I_{\Delta, \mu}(\sigma(t_i)) = I_{\Delta, \mu}(\sigma(t_i))$ for all $i \notin \mu(t)$ by induction hypothesis, and $(I_{\Delta, \mu}(\sigma(t_1)), \ldots, I_{\Delta, \mu}(\sigma(t_k))) \hookrightarrow^{*} \sigma_{\Delta, \mu}(\sigma(t_1)), \ldots, \sigma_{\Delta, \mu}(\sigma(t_k)) = I_{\Delta, \mu}(\sigma(t))$.

- If $t \in \Delta$ or $t$ is non-$\mu$-terminating, then we have that $I_{\Delta, \mu}(\sigma(t)) = I_{\Delta, \mu}(\sigma(t))$ where $t = t'(i_1, \ldots, t_k)$ and $t\cup\mu = t'$.

**Lemma 7.** Let $\tau = (P, R, S, \mu)$ be a CS problem where $P = (G, P)$, $R = (F, R)$ and $S = (H, S)$. Let $\Delta = \{G \cup F \cup H\} \setminus \{\text{root}(t) \mid t \in R \cup t' \in P \cup (R, S, \mu)\}$. If $s$ and $t$ are $\mu$-terminating on $R$ and $s \hookrightarrow_{R, \mu} t$ then $I_{\Delta, \mu}(s) \hookrightarrow^{*}_{\mu(P, R, S, \mu)_{\Delta, \mu}} I_{\Delta, \mu}(t)$.

**Proof.** We proceed by induction on the position $p \in \text{Pos}(s)$ of the redex in the reduction $s \hookrightarrow_{(\rho \cup \mu)} t$. First let $\text{root}(s) \in \Delta$ and $p = \Lambda (t \rightarrow r \in \Delta \setminus \{\text{root}(s)\} \setminus \{\text{root}(r)\})$. Then we have $r \rightarrow^{(\rho \cup \mu)} t$. Let $I_{\Delta, \mu}(s) \hookrightarrow_{(\rho \cup \mu)} I_{\Delta, \mu}(t)$.
Proving Termination in the CSDP Framework

$U^\rho(P,R,S,\mu)$. So we have $s = \sigma(f) \overset{\Delta}{\rightarrow} f(t)$ and $\sigma(r) = t$ for some substitution $\sigma$. Moreover, $r \in T(\Delta,X)$ and all subterms $t'$ of $t$ at active positions are from $T(\Delta,X)$ by the conditions on $\Delta$. By Lemma 6 we get $I_{1,\Delta,\mu}(\sigma(t)) \overset{\mu}{\rightarrow} I_{1,\Delta,\mu}(\sigma(r))$. Now let $\text{root}(s) \in \Delta$ and $p \neq \Delta$.

Hence, $s = t(s_1, \ldots, s_i, \ldots, s_n)$, $t = t(s_1, \ldots, t_i, \ldots, s_n)$, $i \in \mu(t)$, and $s_i \overset{t_i}{\rightarrow} s_i$. The induction hypothesis implies $I_{1,\Delta,\mu}(s_i) \overset{t_i}{\rightarrow} I_{1,\Delta,\mu}(t_i)$ and hence, $I_{1,\Delta,\mu}(s) \overset{t}{\rightarrow} I_{1,\Delta,\mu}(t)$. Finally, we consider the case $\text{root}(s) \notin \Delta$. In this case, $I_{1,\Delta,\mu}(t) \in \text{order}(\Omega_{1,\Delta,\mu}(s) | s \overset{t}{\rightarrow} t)$ because $s \overset{t}{\rightarrow} t$. By applying $\mathcal{C}$ rules, we get $I_{1,\Delta,\mu}(s) \overset{t}{\rightarrow} I_{1,\Delta,\mu}(t)$.

Theorem 10 ($\mu$-Reduction Triple Processor with Usable Rules). Let $\tau = (P,R,S,\mu)$ be a CS problem where $P = (G,P)$, $R = (F,R)$ and $S = (H,S)$.

Let $(\preceq,\succeq)$ be a $\mu$-reduction triple such that

1. $P \subseteq S \cup \preceq$
2. at least one of the following holds:
   (a) $U^\mu(\tau) \subseteq S$ is strongly $\mu$-conservative, $\succeq$ is $\mathcal{C}$-compatible,
   (b) $U^\mu(\tau) \subseteq S, \preceq$ is $\mathcal{C}$-compatible,
   (c) $R \subseteq \preceq$.
3. and, whenever $P_X \neq \emptyset$ we have that $S \subseteq \preceq \cup \preceq \cup \preceq$.

Let $P_\preceq = \{u \rightarrow v \in P \mid u \succeq v\}$ and $S_\preceq = \{s \rightarrow t \in S \mid s \preceq t\}$. Then, the processor $\text{Proc}_{LR}$ given by

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

is sound and complete.

Proof. Completeness is obvious, since $P \setminus P_\preceq \subseteq P$ and $S \setminus S_\preceq \subseteq P$. Regarding soundness, we proceed by contradiction. Assume that there is an infinite minimal $(P,R,S,\mu)$-chain $A$, but there is no infinite minimal $(P \setminus P_\preceq, R, S \setminus S_\preceq, \mu)$-chain. Due to the finiteness of $P$ and $S$, we can assume that there are subsets $Q \subseteq P$ and $T \subseteq S$ such that $A$ has a tail $B$

$$\sigma(u_1) \overset{t_1}{\rightarrow} \cdots \overset{t_n}{\rightarrow} \sigma(u_n)$$

for some substitution $\sigma$, where all pairs in $Q$ and all rules in $T$ are infinitely often used (note that, if $T \neq \emptyset$, then $T \neq \emptyset$ and $Q_X \neq \emptyset$), and, for all $i \geq 1$, (1) if $u_i \rightarrow v_i \in Q_X$, then $t_i = \sigma(v_i)$ and (2) if $u_i \rightarrow v_i = u_i \rightarrow x_i \in Q_X$, then $\sigma(u_i) \overset{t_i}{\rightarrow} \sigma(x_i) \overset{A_{T_X,i}}{\rightarrow} \sigma(x_i) \overset{A_{T,i}}{\rightarrow} \sigma(u_{i+1})$. Moreover, all $t'_i$ are $\mu$-terminating.

First, we consider Item 2a. If $P \cup U^\mu(P,R,S,\mu)$ is strongly conservative, we have no collapsing pairs, that is, we can write:

$$\sigma(u_1) \overset{t_1}{\rightarrow} \cdots \overset{t_n}{\rightarrow} \sigma(u_n)$$
and we do not use the rules in $S$. We apply $I_{0,\Delta,\mu}$ in Definition 18 to the initial term. Let $\Delta = (G \cup F) \setminus \{\text{root}(t) \mid t \in \mathcal{R} \setminus \mathcal{U}^*(P, R, S, \mu)\}$. Note that the application of $I_{0,\Delta,\mu}$ is always possible since all terms $\sigma(v_i)$ and $\sigma(u_{i+1})$ are $\mu$-terminating due to the minimality of the chain. Using Lemma 4, we obtain $I_{0,\Delta,\mu}(\sigma(v_i)) \vdash I_{\sigma(v_i)}^+(P, R, S, \mu; C_\prec, \mu)$ for all variable subterms $v_i$ at $\mu$-replacing $v_i$. By Proposition 3 and Lemma 4, we have $\sigma_{I_{0,\Delta,\mu}}(v_i) = \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$ for all $i \geq 1$. Moreover, by the definition of $\mathcal{U}^*(P, R, S, \mu)$, for all nonvariable subterms $v_i'$ at $\mu$-replacing positions of $v_i$, we have $\text{root}(v_i') \in \Delta$. By Proposition 3 and Lemma 4, we have $\sigma_{I_{0,\Delta,\mu}}(v_i) = \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$ for all $i \geq 1$. Moreover, if $\sigma_{I_{0,\Delta,\mu}}(u_{i+1}) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$, then $\sigma_{I_{0,\Delta,\mu}}(u_{i+1}) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Since $\sigma_{I_{0,\Delta,\mu}}(u_{i+1}) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$, then $\sigma_{I_{0,\Delta,\mu}}(u_{i+1}) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$.

No pair $u \rightarrow v \in \mathcal{Q}$ satisfies $u \not\succ v$. Otherwise, we get a contradiction by considering that, if $u_i \rightarrow v_i \in \mathcal{Q}$, then $\sigma_{I_{0,\Delta,\mu}}(v_i) \vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Since $\sigma_{I_{0,\Delta,\mu}}(v_i) \vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$, then $\sigma_{I_{0,\Delta,\mu}}(v_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Therefore, $B$ is an infinite minimal $(P \setminus \mathcal{Q}, R, S, \mu)$-chain, leading to a contradiction.

Now, we consider Item 2b. If $P \cup \mathcal{U}^*(P, R, S, \mu)$ is not strongly conservative, we apply $I_{1,\Delta,\mu}$ in Definition 20 to the initial term. We let $\Delta = (G \cup F \cup H) \setminus \{\text{root}(t) \mid t \in \mathcal{R} \setminus \mathcal{U}^*(P, R, S, \mu)\}$. We consider two cases:

1. If $u_i \rightarrow v_i \in \mathcal{Q}$, then by Lemma 7 we get $I_{1,\Delta,\mu}(\sigma(v_i)) \vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Since $\sigma_{I_{0,\Delta,\mu}}(v_i) \vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$, then $\sigma_{I_{0,\Delta,\mu}}(v_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Therefore, $\sigma_{I_{0,\Delta,\mu}}(v_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$.

Since $u_i \rightarrow v_i \in \mathcal{Q}$, by stability of $\succ$ and $\equiv$, we have $\sigma_{I_{0,\Delta,\mu}}(u_i) \vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$ for all $i \geq 1$.

No pair $u \rightarrow v \in \mathcal{Q}$ satisfies $u \not\equiv v$ and no rule $\ell \rightarrow r \in \mathcal{T}$ satisfies $\ell \equiv t$.

Otherwise, we get a contradiction by considering the following two cases:

1. If $u_i \rightarrow v_i \in \mathcal{Q}$, then $\sigma_{I_{0,\Delta,\mu}}(v_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Since $\sigma_{I_{0,\Delta,\mu}}(v_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$, by using transitivity of $\succ$ and compatibility between $\succ$ and $\equiv$, we conclude that $\sigma_{I_{1,\Delta,\mu}}(u_i) \not\vdash \sigma_{I_{\sigma(v_i)}}^+(P, R, S, \mu; C_\prec, \mu)$. Therefore, $B$ is an infinite minimal $(P \setminus \mathcal{Q}, R, S, \mu)$-chain, leading to a contradiction.
2. If \( u_i \rightarrow v_i = u_i \rightarrow x_i \in Q_X \) (which is not empty whenever \( T \) is not empty),

then \( \sigma_{T_\Delta,\Delta}(v_i) = \sigma_{T_\Delta,\Delta}(x_i) \). Since \( \ell_j(\geq \cup \geq \cup \cap \ell) \) for all \( \ell_j \rightarrow r_j \in T_{\Delta,j} \),

we have \( \sigma_{T_\Delta,\Delta}(v_i) = \sigma_{T_\Delta,\Delta}(x_i) \). Furthermore, we are assuming that \( \ell \geq \cup \geq \cup \cap r \) for all \( \ell \rightarrow r \in T_\ell \).

Since \( \sigma_{T_\Delta,\Delta}(\ell_i) \rightarrow r_i \sigma_{T_\Delta,\Delta}(r_i) \), for some \( \ell_i \rightarrow r_i \in T_\ell \) and, by stability of the (quasi-)orderings we have that \( \sigma_{T_\Delta,\Delta}(\ell_i) \) (\( \geq \cup \geq \cup \cap \sigma_{T_\Delta,\Delta}(r_i) \)). Hence, by transitivity of \( \geq \) (and compatibility of \( \geq \) and \( \geq \)), we have \( \sigma_{T_\Delta,\Delta}(v_i) = \sigma_{T_\Delta,\Delta}(x_i) \). Hence, \( \sigma_{T_\Delta,\Delta}(r_i) \geq \sigma_{T_\Delta,\Delta}(x_i) \). Therefore, again by transitivity of \( \geq \) and compatibility of \( \geq \), we conclude that \( \sigma_{T_\Delta,\Delta}(v_i) \geq \sigma_{T_\Delta,\Delta}(x_i) \) and hence \( \sigma_{T_\Delta,\Delta}(u_i) \).

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 \( \sigma_{T_\Delta,\Delta}(u_i) \) for all \( i \in \mathcal{I} \). Thus, by using the compatibility conditions of the \( \mu \)-reduction triple, we obtain an infinite decreasing \( \geq \) sequence which contradicts well-foundedness of \( \geq \).

Therefore, \( B \) is an infinite minimal \( (P \setminus P_r, R, S \setminus S_r, \mu) \)-chain, thus leading to a contradiction.

Finally, Item 2c is proved in Theorem 9.

J Proof of Theorem 11

Theorem 11 (Subterm Processor). Let \( \tau = (P, R, S, \mu) \) be a CS problem

where \( R = (P, R) = (C \cup D, R) \), \( P \) and \( S \). Assume that (1) \( \text{Root}(P) \cap D = \emptyset \), (2) the rules in \( P \) are noncollapsing, and (3) if \( X \in P \neq \emptyset \),

then for all \( X \neq s \in S \), \( \text{root}(t) \in \text{Root}(P) \). Let \( \pi \) be a simple projection for \( P \).

Let \( S_\pi = \{ s \rightarrow \pi(t) \mid s \rightarrow t \in S_\pi \} \). Let \( P_\pi, S_\pi = \{ u \rightarrow v \in P \mid \pi(u) \cap \pi(v) \} \) and \( S_\pi, D_\pi = S_\pi \cup \{ s \rightarrow t \in S_\pi \mid s \neq \pi(t) \} \). Then, \( \text{Proc}_{s\text{-subterm}} \) given by

\[
\text{Proc}_{s\text{-subterm}}(\tau) = \begin{cases} 
\{(P \setminus P_{s \in D_\pi}, R, S \setminus S_{s \in D_\pi}, \mu) \} & \text{if } \pi(P) \subseteq \Delta \mu \\
\{(P, R, S, \mu)\} & \text{otherwise}
\end{cases}
\]

is sound and complete.

Proof. Completeness is obvious because \( P \setminus P_{s \in D_\pi} \subseteq P \) and \( S \setminus S_{s \in D_\pi} = S \). For soundness, we proceed by contradiction. Assume that there is an infinite minimal \( (P, R, S, \mu) \)-chain \( A \) but there is no infinite minimal \((P \setminus P_{s \in D_\pi}, R, S \setminus S_{s \in D_\pi}, \mu)\)-chain. Since \( P \) and \( S \) are finite, we can assume that there are subsets \( Q \subseteq P \) and \( T \subseteq S \) such that \( A \) has a tail \( B \) which is an infinite minimal \((Q, R, T, \mu)\)-chain where all pairs in \( Q \) and rules in \( T \) are infinitely often used. Note that, if \( T \neq \emptyset \), then \( \emptyset \neq T \) and \( Q_X \neq \emptyset \). Assume that \( B \) is as follows:

\[
\sigma(u_1) \overset{\Delta}{\rightarrow} \cdots \sigma(u_2) \overset{\Delta}{\rightarrow} \cdots
\]
for some substitution $\sigma$, and, for all $i \geq 1$, (A) if $u_i \rightarrow v_i \in QG$, then $t_i = \sigma(v_i)$ and (B) if $u_i \rightarrow v_i = u_i \rightarrow x_i \in QX$ and $t_i \rightarrow r_i \in T_{\mu}$, then $\sigma(u_i) \rightarrow_\mu \sigma(v_i)$. Note that, for all $i \geq 1$,

- $\text{root}(\sigma(u_i)) \in \text{Root}(P)$ because $\text{root}(u_i) \in \text{Root}(P)$,
- if $v_i \rightarrow v_i \in P_G$, then $t_i = \sigma(v_i)$. By (2), $v_i \notin X$. Therefore, $\text{root}(v_i) \in \text{Root}(P)$,
- if $u_i \rightarrow v_i \in P_X$, then $t_i = \sigma(v_i)$. By (3), $\text{root}(v_i) \in \text{Root}(P)$. Therefore, $\text{root}(t_i) \in \text{Root}(P)$.

Since $t_i \rightarrow_\mu \sigma(u_{i+1})$ for all $i \geq 0$ and $\text{Root}(P) \cap D = \emptyset$ (due to (1)), we can actually write $t_i \rightarrow_\mu \sigma(u_i)$ because $\mu$-rewritings with $\mathcal{R}$ cannot change $\text{root}(t_i)$. Hence $x_i \rightarrow_{\mu} \sigma(x_i)$ and also $\text{root}(t_i) = \text{root}(\sigma(u_{i+1}))$ for all $i \geq 1$. Since $\pi(u_i) \in P_\mu \pi(v_i)$ for all $i \geq 1$, by stability of $P_\mu$, we have, for all $i \geq 1$

- if $u_i \rightarrow v_i \in Q_G$, then $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \in P_\mu \pi(\sigma(v_i)) = \pi(\sigma(v_i)) = \pi(t_i)$,
- if $u_i \rightarrow x_i \in Q_X$, then $\pi(\sigma(x_i)) \rightarrow_\mu \pi(x_i) \rightarrow_\mu \pi(t_i)$. We prove by induction on

the length $m$ of the sequence $\sigma(x_i) \rightarrow_\mu \mu \pi(x_i)$ that $\sigma(x_i) \geq_\mu \pi(t_i)$,

- if $m = 0$ then $\sigma(x_i) = t_i$,
- if $m > 1$, then we can write $\sigma(x_i) \rightarrow_\mu \mu \pi(x_i) \rightarrow_\mu \pi(x_i)$ where $\pi(x_i)$ is obtained from $\sigma(x_i)$ in $m-1$ $\mu$-steps. By the induction hypothesis, $\sigma(x_i) \geq_\mu \pi(x_i) = \pi(t_i)$ for some $t_i \rightarrow r_i \in T_{\mu} \subseteq S_{\mu}$. Since $t_i \rightarrow r_i$ (by definition of $S_{\mu}$), then by stability of $\mu$, we have $\pi(t_i) \in P_\mu \pi(r_i)$ and, hence, $\pi(t_i) \in S_{\mu}$. Since $\pi(t_i) \in S_{\mu}$, we have that $t_i = \sigma(r_i)$. Since $\text{root}(t_i) \in \text{Root}(P)$, and $\mu \subseteq P_{\mu} \pi(r_i)$ (by the assumption $S_{\mu} \subseteq P_{\mu}$), by stability of $P_{\mu}$, we have $t_i = \sigma(t_i) = \pi(\sigma(r_i)) = \pi(t_i)$ and, hence, $\sigma(\pi(u_i)) = \sigma(\pi(u_i)) \in P_\mu \sigma(\pi(x_i)) = \sigma(x_i) \in P_\mu \sigma(\pi(t_i)) = \pi(t_i)$.

Therefore, we have $\pi(\sigma(u_i)) \in P_\mu \pi(t_i)$ in any case. No pair $u \rightarrow v \in Q$ satisfies that $\pi(u) \in P_\mu \pi(v)$ and no rule $t \rightarrow r \in T$ satisfies $\pi(t) \in P_\mu \pi(r)$. Otherwise, we get a contradiction in both of the following two complementary cases:

1. if $b(t) \notin P(t)$ for all $t \in P(Q)$, 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 $\pi(\sigma(u_{i+1})) \supseteq P_\mu \pi(\sigma(x_i))$, we have that $\pi(t_i) \supseteq P_\mu \pi(\sigma(s_{i+1}))$ for all $i \geq 0$. Furthermore, since we assume $\pi(u) \in P_\mu \pi(v)$ for some $u \rightarrow v \in Q$, $t \rightarrow r$ for some $t \rightarrow r \in T_{\mu}$, or $t \rightarrow r \in T_{\mu}$ which apply infinitely often in $B$ (remind that $T = \emptyset$ if and only if $Q = \emptyset$), and by stability of $P_\mu$, there is a maximal infinite set $J = \{j_1, j_2, \ldots\} \subseteq N$ such that $\pi(t_{j_i}) \supseteq P_\mu \pi(x_{j_i+1})$ for all $i \geq 1$. Thus, we obtain an infinite sequence $\pi(t_{j_1}) \supseteq P_\mu \pi(t_{j_2}) \supseteq P_\mu \cdots$ which contradicts the well-foundedness of $P_\mu$. 


2. if \( \pi(f) \in \mu(f) \) for some \( f \in \text{Root}(Q) \), then, since \( \text{root}(t_i) = \text{root}(\sigma(u_{i+1})) \) and all pairs in \( Q \) occur infinitely often in \( B \), we can assume that \( \text{root}(t_i) = f \). Furthermore, since \( A \) is minimal, we can assume that \( t_i \) is \( \mu \)-terminating (w.r.t. \( R \)). Since \( \pi(t_i) \overset{R}{\Rightarrow} \pi(\sigma(u_{i+1})) \) and \( \pi(\sigma(u_{i+1})) \overset{\mu}{\Rightarrow} \pi(t_{i+1}) \) for all \( i \geq 0 \), the sequence \( B \) is transformed into an infinite \( \overset{R}{\Rightarrow} \mu \)-sequence

\[
\pi(t_1) \overset{R}{\Rightarrow} \pi(\sigma(u_2)) \overset{\mu}{\Rightarrow} \pi(t_2) \overset{R}{\Rightarrow} \pi(\sigma(u_3)) \overset{\mu}{\Rightarrow} \pi(t_3) \overset{R}{\Rightarrow} \cdots
\]

containing infinitely many \( \overset{R}{\Rightarrow} \mu \)-steps, due to \( \pi(u) \overset{R}{\Rightarrow} \pi(v) \) for some \( u \sim r \in Q \), \( t \overset{\mu}{\Rightarrow} r \) for some \( t \sim r \in T \), or \( t \overset{\mu}{\Rightarrow} r \) for some \( t \sim r \in T \) which apply infinitely often in \( B \). Since \( \overset{R}{\Rightarrow} \mu \) is well-founded, the infinite sequence must also contain infinitely many \( \overset{R}{\Rightarrow} \mu \)-steps. By making repeated use of the fact that \( \overset{R}{\Rightarrow} \mu \circ \overset{R}{\Rightarrow} \mu \overset{R}{\Rightarrow} \mu \), we obtain an infinite \( \overset{R}{\Rightarrow} \mu \)-sequence starting from \( \pi(t_1) \). Thus, \( \pi(t_1) \) is not \( \mu \)-terminating with respect to \( R \). Since \( \pi(f) \in \mu(f) \) and hence \( t_i \overset{R}{\Rightarrow} \mu(t_i) \), this implies that \( t_i \) is not \( \mu \)-terminating.

This contradicts \( \mu \)-termination of \( t_i \).

Therefore, \( Q \subseteq P \setminus P_{\pi,\overset{R}{\Rightarrow} \mu} \) and \( T \subseteq S \setminus S_{\pi,\overset{R}{\Rightarrow} \mu} \). Hence, \( B \) is an infinite minimal \((Q,R,T,\mu)\)-chain. This contradicts our initial argument.

K Proof of Theorem 12

Theorem 12 (Non-\( \mu \)-Replacing Projection Processor). Let \( \tau = (P,R,S,\mu) \) be a CS problem where \( \tau = (C \cup D,R) \) and \( \tau = (\tilde{G},P) \). Assume that (1) \( \text{Proof of Theorem } 12 \)

1. for all \( f \in \text{Root}(P) \), \( \pi(f) \notin \mu(f) \),
2. \( \pi(P) \subseteq \geq \), and,
3. whenever \( S \neq \emptyset \) and \( P_X \neq \emptyset \), we have that \( S_X \subseteq \geq \).

Let \( S_{\pi,\overset{R}{\Rightarrow} \mu} = \{ s \sim t \in P \mid \pi(u) \overset{R}{\Rightarrow} \pi(v) \} \) and \( S_{\pi,\overset{R}{\Rightarrow} \mu} = \{ s \sim t \in S_{\overset{R}{\Rightarrow} \mu} \mid s > t \} \cup \{ s \sim t \in S_{\overset{R}{\Rightarrow} \mu} \mid s < t \} \). Then, the processor \( \text{Proc}_{\text{NRP}} \) given by

\[
\text{Proc}_{\text{NRP}}(P,R,S,\mu) = \begin{cases} \{ (P \setminus P_{\pi,\overset{R}{\Rightarrow} \mu},R,S \setminus S_{\pi,\overset{R}{\Rightarrow} \mu}) \} & \text{if } (1), (2), \text{ and } (3) \text{ hold} \\
\{ (P,\mu) \} & \text{otherwise} 
\end{cases}
\]

is sound and complete.

Proof. Completeness is obvious because \( P \setminus P_{\pi,\overset{R}{\Rightarrow} \mu} \subseteq P \) and \( S \setminus S_{\pi,\overset{R}{\Rightarrow} \mu} \subseteq S \). For soundness, we proceed by contradiction. Assume that there is an infinite minimal \((P,R,S,\mu)\)-chain \( A \) but there is no infinite minimal \((P \setminus P_{\pi,\overset{R}{\Rightarrow} \mu},R,S \setminus S_{\pi,\overset{R}{\Rightarrow} \mu})\)-chain. Since \( P \) and \( S \) are finite, we can assume that there are subsets \( Q \subseteq P \) and \( T \subseteq S \) such that \( A \) has a tail \( B \) which is an infinite minimal \((Q,R,T,\mu)\)-chain
where all pairs in $Q$ and rules in $T$ are infinitely often used. Note that, if $T \neq \emptyset$, then $T_2 \neq \emptyset$ and $Q_X \neq \emptyset$. Assume that $B$ is as follows:

$$(\sigma(u_1) \leftarrow_{\mathcal{R}, \mu} \sigma(u_2) \leftarrow_{\emptyset} \cdots)$$

for some substitution $\sigma$, and, for all $i \geq 1$, (A) if $u_i \rightarrow v_i \in Q_0$, then $t_i = \sigma(v_i)$ and (B) if $u_i \rightarrow v_i = u_i \rightarrow x_i \in Q_X$ and $t_i \rightarrow r_i \in T_i$, then $\sigma(u_i) \leftarrow_{\emptyset} \sigma(v_i)$ and $\sigma(x_i) \leftarrow_{\mathcal{T}_\nu, \mu} \sigma(r_i) = t_i$. Note that, for all $i \geq 1$,

- $\operatorname{root}(\sigma(u_i)) \in \operatorname{Root}(P)$ because $\operatorname{root}(u_i) \in \operatorname{Root}(P)$,
- If $u_i \rightarrow v_i \in P_Q$, then $t_i = \sigma(v_i)$. By (2), $v_i \notin X$. Therefore, $\operatorname{root}(v_i) \in \operatorname{Root}(P)$,
- If $u_i \rightarrow v_i \in P_X$, then $t_i = \sigma(v_i)$. By (3), $\operatorname{root}(v_i) \in \operatorname{Root}(P)$. Therefore, $\operatorname{root}(t_i) \in \operatorname{Root}(P)$.

Since $t_i \leftarrow_{\mathcal{R}, \mu} \sigma(u_{i+1})$ for all $i \geq 0$ and $\operatorname{Root}(P) \cap D = \emptyset$ (due to (1)), we can actually write $t_i \leftarrow_{\mathcal{R}, \mu} \sigma(u_{i+1})$ because $\mu$-rewritings with $\mathcal{R}$ cannot change $\operatorname{root}(t_i)$ and $\operatorname{root}(t_i) = \operatorname{root}(\sigma(u_{i+1}))$. Since $\pi(u_i) \geq \pi(v_i)$ for all $i \geq 1$, by stability of $\mathcal{Z}$, we have, for all $i \geq 1$

- If $u_i \rightarrow v_i \in Q_0$, then $\pi(\sigma(u_i)) = \pi(\sigma(v_i)) \geq \pi(\sigma(v_i)) = \pi(t_i)$.
- If $u_i \rightarrow x_i \in Q_X$, then $\pi(\sigma(x_i)) \leftarrow_{\mathcal{T}_\nu, \mu} t_i' \leftarrow_{\mathcal{T}_\nu, \mu} t_i$.

We prove by induction on the length $m$ of the sequence $\sigma(x_i) \leftarrow_{\mathcal{T}_\nu, \mu} t_i'$ that $\sigma(x_i) \geq t_i'$.

- If $m = 0$ then $\sigma(x_i) = t_i'$.
- If $m > 1$, then we can write $\sigma(x_i) \leftarrow_{\mathcal{T}_\nu, \mu} t_i' \leftarrow_{\mathcal{T}_\nu, \mu} t_i''$ where $t_i''$ is obtained from $\sigma(x_i)$ in $m-1$ $\mathcal{T}_\nu, \mu$-steps. By the induction hypothesis, $\sigma(x_i) \geq t_i'' = \sigma(t)$ for some $t \rightarrow r \in T_\nu \subseteq T_\mu$. Since $t \geq r$ (because $S_p \subseteq S_\nu \subseteq S_\mu$), then by stability of $\mathcal{Z}$ we have $\sigma(t) \geq \sigma(r) = t_i'$ and, hence, $\sigma(x_i) \geq t_i'$.

Since $t_i' = \sigma(t_i)$, we have that $t_i = \sigma(r_i)$. Since $\operatorname{root}(t_i) \in \operatorname{Root}(P)$, and $\ell_i \geq \pi(r_i)$ (by the assumption $S_p \subseteq S_\mu$), by stability of $\mathcal{Z}$, we have $t_i' = \sigma(t_i) \geq \sigma(\pi(r_i)) = \pi(\sigma(r_i)) = \pi(t_i)$ and, hence,

$\pi(\sigma(u_i)) = \pi(\sigma(v_i)) \geq \pi(\sigma(v_i)) = \sigma(x_i) \geq \sigma(t_i) \geq \sigma(\pi(r_i)) = \pi(t_i)$. 

Therefore, we have $\pi(\sigma(u_i)) \geq \pi(t_i)$ in any case. No pair $u \rightarrow v \in Q$ satisfies that $\pi(u) > \pi(v)$, no rule $\ell \rightarrow r \in T_\mu$ satisfies that $\ell > r$, and no rule $\ell \rightarrow r \in T_\nu$ 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 $f \in \operatorname{Root}(Q)$, 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 $\pi(u_i) \geq \pi(v_i)$ and by stability of $\mathcal{Z}$, we have that $\pi(\sigma(u_i)) = \sigma(\pi(u_i)) \geq \pi(\sigma(v_i)) = \pi(v_i)$. Now, we distinguish two cases:
(a) If $u_i \rightarrow v_i \in \mathcal{Q}_G$, then $\pi(t_i) = \pi(\sigma(v_i)) = \sigma(\pi(v_i))$. Thus, $\pi(\sigma(u_i)) \succeq \pi(t_i)$.

(b) If $u_i \rightarrow v_i \in \mathcal{Q}_X$, then $\sigma(\pi(v_i)) = \sigma(x_i)$. Thus, $\sigma(x_i) \succeq \sigma(t_i)$ and $\sigma(t_i) \succeq \sigma(\pi(r_i)) = \pi(\sigma(r_i)) = \pi(t_i)$ (by (3)). Thus, $\pi(\sigma(u_i)) \succeq \pi(t_i)$.

Thus, we always have $\pi(\sigma(u_i)) \succeq \pi(t_i)$. We obtain an infinite $\succeq$ sequence

$$
\pi(\sigma(u_1)) \succeq \pi(t_1) = \pi(\sigma(u_2)) \succeq \pi(t_2) \cdots
$$

Since pairs in $\mathcal{Q}$ and rules in $T$ occur infinitely often, this sequence contains infinitely many $\succ$ steps starting from $\pi(\sigma(u_1))$. This contradicts the well-foundedness of $\succ$.

Therefore, $\mathcal{Q} \subseteq \mathcal{P} \setminus \mathcal{P}_{\succ}$ and $T \subseteq \mathcal{S} \setminus \mathcal{S}_{\succ}$, i.e., $B$ is an infinite minimal $(\mathcal{P} \setminus \mathcal{P}_{\succ}, \mathcal{R}, \mathcal{S} \setminus \mathcal{S}_{\succ}, \mu)$-chain. This contradicts our initial assumption.