DISSERTATION

Applications and Generalizations of Context-Sensitive Term Rewriting

ausgeführt zum Zwecke der Erlangung des akademischen Grades eines Doktors der technischen Wissenschaften unter der Leitung von

Ao.Univ.Prof. Dipl.-Inf. Dr.rer.nat. Bernhard Gramlich

E185/2
Institut für Computersprachen
Arbeitsbereich Theoretische Informatik und Logik

eingereicht an der Technischen Universität Wien
Fakultät für Informatik

von

Dipl.-Ing. Felix Schernhammer

Matrikelnummer: 0225493
Leonard Bernsteinstraße 8/1/8.5, 1220 Wien

This work has been supported by the Austrian Academy of Sciences within the DOC program under grant number 22.361.

Acknowledgments

First of all I would like to thank my PhD advisor Prof. Bernhard Gramlich for his ongoing support and confidence in me. He introduced me to the world of research and provided me with the freedom needed to persue my ideas. Furthermore, I would like to express my gratitude to the Austrian Academy of Science for granting me the scholarship that made this work possible in the first place.

Special gratitude is dedicated to Prof. José Meseguer and his group at the University of Illinois at Urbana-Champaign for their hospitality, support and friendship during my research visit in the USA. I would also like to thank all of my colleges of the Theory and Logic Group at the Vienna University of Technology who helped to broaden my scientific horizon in many seminars and provided me with guidance and help whenever needed.

Finally, I would like to thank my family and friends for their ongoing love and support.

Felix Schernhammer

Abstract

Term rewriting is a formalism that enables us to efficiently deal with many problems related to symbolic equations. It is particularly used in fields like equational programming, formal verification of software and automated deduction. The simple idea behind term rewriting is to replace equals by equals in an object (in our case terms) until a form is reached, which represents the result of the computation. In an operational sense the formalism of term rewriting does not suggest how this replacement is carried out. Thus, in general rewrite derivations (as we call sequences of replacements) are highly non-deterministic. Especially in practical fields like equational programming and executable specifications it is important to have means to guide rewrite derivations and thus to reduce their nondeterminism in order to make them more efficient and/or produce the desired results. One major approach to achieve this is context-sensitivity. The idea of context-sensitive rewriting is to exploit the term structure of the objects in order to identify positions at which replacements are allowed and others where no replacements may occur.

In the first part of this thesis we investigate the benefit one gains from using traditional notions of context-sensitivity in the simulation of conditional rewrite systems by unconditional ones. Conditional rewriting is an extension of ordinary rewriting where rewrite rules are guarded by conditions. These conditional rules may only be used for computations if the conditions are satisfied. It turns out that the use of context-sensitivity significantly improves the accuracy one can achieve in this simulation. Indeed, we show that by using a concrete transformation from conditional rewrite system into unconditional context-sensitive ones, the computational power of the conditional and transformed system is entirely the same. Hence, the accuracy of a simulation of conditional rewriting by using this transformation is arguably optimal. Besides simulation, we use this transformation to derive criteria for the practically crucial properties of operational termination and $C_{\mathcal{E}}$-operational termination of conditional rewrite systems based on (local) termination properties of the transformed unconditional rewrite systems. The latter properties are easier to verify and an extensive amount of existing research deals with verifying these properties automatically.

The second part of the thesis introduces a new approach to context-sensitivity in term rewriting. The basic idea is to define term patterns, which are called "forbidden patterns", that determine parts of a term that are forbidden for reduction whenever they match the term. This new formalism is more general than many existing approaches to context-sensitivity. Nevertheless, the definition of rewriting with forbidden patterns is comparatively simple, thus making the approach feasible in practice despite its expressiveness. We develop a basic infrastructure of results concerning rewriting with forbidden patterns, providing in particular criteria for completeness, confluence and termination. Moreover, we present a method to automatically infer sets of forbidden patterns for given rewrite systems that impose desirable restrictions on the induced rewrite relation.

Finally, in the third part of the thesis, we describe the termination laboratory VMTL, which has been developed to evaluate the above as well as various other methods for the automated termination analysis of conditional rewrite systems as well as of rewrite systems with forbidden patterns introduced in this thesis.

Zusammenfassung

Termersetzung ist ein Formalismus, der in vielen Gebieten der theoretischen Informatik verwendet wird, in denen symbolische Gleichungen bzw. symbolische Berechnungen eine Rolle spielen. Sie findet insbesondere in den Gebieten der gleichungsbasierten Programmierung, der formalen Verifikation von Software und im automatischen Beweisen ihre Anwendung. Die prinzipielle Idee hinter Termersetzung ist Terme zu vereinfachen, indem Teilterme durch einfachere, aber semantisch gleiche Terme ersetzt werden. Da es a priori unklar ist welche Teilterme zuerst vereinfacht werden sollen, ist der Vorgang des Termersetzens hochgradig nichtdeterministisch. In der Praxis spielt die Reihenfolge, in der Vereinfachungen vorgenommen werden, allerdings oft eine große Rolle, beispielsweise wenn bestimmte Vereinfachungssequenzen wesentlich schneller zum gewünschten Resultat führen als andere, oder bestimmte problematische Vereinfachungssequenzen zu Nichtterminierung führen. Deshalb versucht man den Nichtdeterminismus der Termersetzung durch Ersetzungsstrategien zu verringern oder zu eliminieren. Ein wichtiger konkreter Ansatz für die Definition solcher Strategien ist kontextsensitives Termersetzen.

Im ersten Teil der Dissertation wird kontextsensitives Termersetzen verwendet, um Reduktionen bezüglich bedingter Termersetzungssysteme zu simulieren. In bedingten Termersetzungssystemen werden für jede Ersetzungsregel Bedingungen definiert, unter denen die Regel anwendbar ist. Durch die Verwendung von kontextsensitivem Termersetzen ist es tatsächlich möglich, bedingte Termersetzung akkurat durch unbedingte kontextsensitive zu simulieren. Die Dissertation enthält eine Transformation, die bedingte Termersetzungssysteme in unbedingte kontextsensitive übersetzt, wobei nach der Übersetzung exakt die selben Resultate berechnet werden können wie davor. Diese Transformation wird in weiterer Folge dazu benutzt, um praktisch relevante Eigenschaften von bedingten Ersetzungssystemen wie zum Beispiel operationale Terminierung oder $C_{\mathcal{E}}$-operationale Terminierung, zu verifizieren, indem man Terminierungseigenschaften des transformierten unbedingten Systems nachweist. Dieses Vorgehen ermöglicht es, existierende Methoden für die Terminierungsanalyse von unbedingten Ersetzungssystemen wiederzuverwenden.

Im zweiten Teil der Dissertation wird ein neuer Ansatz zu eingeschränktem Termersetzen eingeführt, der kontextsensitives Termersetzen verallgemeinert. In diesem Ansatz werden sogenannte “Forbidden Patterns” verwendet, um Teile eines Terms zu identifizieren, in denen keine Vereinfachungen durchgeführt werden dürfen. Viele wichtige Standardreduktionsstrategien für Termersetzung sind als Spezialfall dieses neuen Ansatzes vergleichsweise einfach beschreibbar. Trotz dieser Allgemeinheit beruht Termersetzung mit “Forbidden Patterns” auf vergleichsweise einfachen und gut handhabbaren Grundprinzipien. Dadurch ist der Ansatz relativ einfach zu verwenden und die Analyse von Ersetzungssystemen mit “Forbidden Patterns” in der Praxis effizient möglich. Die Dissertation enthält Kriterien für Konfluenz, Terminierung und Vollständigkeit für Termersetzung mit “Forbidden Patterns”. Außerdem wird gezeigt, wie man für ein gegebenes Termersetzungssystem geeignete “Forbidden Patterns” automatisch generieren kann.

Im dritten Teil der Dissertation wird das Softwaretool VMTL vorgestellt, das Termersetzungssysteme automatisch auf Terminierung untersucht. Das Tool enthält u.a. Implementierungen aller Methoden zur Terminierungsanalyse, die in der Dissertation vorgestellt werden und unterstützt insbesondere bedingte Systeme und “Forbidden Patterns”.

Contents

Chapter 1

Introduction

1.1 Term Rewriting

Term rewriting is a formal concept for dealing with various kinds of problems related to equations or state transitions. Such problems typically arise in areas like automated deduction (e.g., [54]), functional and functional logic programming (e.g., [67, 43]) or formal verification of software (e.g., [66]). The basic idea of term rewriting is to simplify (rewrite, reduce) given expressions (terms) according to specified rules (the rewrite system). These rewrite rules may contain variables that can be arbitrarily instantiated, hence a rewrite rule is actually a schema abbreviating a (usually infinite) set of actual simplifications it describes. Moreover, simplifications may be performed at any position in a given term. The latter two properties are called *stability under substitutions* respectively *stability under contexts* of rewriting. They are also called *closure properties.* Note that the notion of simplification or reduction is actually a bit misleading since depending on the used rewrite rules terms might not be syntactically simplified through rewriting.

Example 1. *Consider the following two rewrite rules.*

$$\begin{aligned} s(x) + y &\rightarrow s(x+y) \\ 0 + y &\rightarrow y \end{aligned}$$

The rules contain two variables x and y. Term $s(s(0)) + s(s(0))$ can be simplified to $s(s(s(s(0))))$ by applying the first rule twice and second rule once, i.e.,

$$\underline{s(s(0)) + s(s(0))} \rightarrow s(\underline{s(0) + s(s(0))}) \rightarrow s(s(\underline{0 + s(s(0))})) \rightarrow s(s(s(s(0))))$$

In the above reduction sequence, the subterms that are actually simplified by the rewrite rules are underlined. These subterms are instances of the left-hand sides of one of the given rewrite rules. Such instances are called redexes. *Note that when we interpret the "+" symbol as addition, "0" as zero and the "s" symbol as the successor function on natural numbers (i.e., s(x) = x + 1), then one can use the above rewrite* *rules to recursively compute the sum of two unary encoded (i.e., $n = s^n(0)$) natural numbers. By interpreting the given reduction sequence in this way, we have computed the sum of $2 = s(s(0))$ and 2 which is $4 = s(s(s(s(0))))$. Note that the final term $s(s(s(s(0))))$ cannot be simplified any further. We call such terms* normal forms *(of the initial term).*

Despite the interpretation of the rewrite rules in Example 1, term rewriting is a purely syntactical transformation on terms, so the terms, respectively the symbols they are composed of, do not have any meaning a priori. This purely syntactic character of term rewriting and thus its simplicity is one of the reasons why term rewriting (and variations like narrowing) have successfully been applied in many important areas of computer science.

An important aspect of the concept of term rewriting is its inherent nondeterminism concerning the order of simplifications. This is to say that terms can be simplified in several ways. Thus, any concrete simplification has to repeatedly choose among various possible simplification steps. However, these choices can have extensive consequences regarding the efficiency and even effectiveness of the overall simplification process.

Example 2. *Consider the following rewrite rules*

$$\begin{aligned} from(x) &\to x : from(s(x)) && (1.1)\\ take(s(x), y : ys) &\to y : take(x, ys) && (1.2)\\ take(0, y) &\to nil\\ take(x, nil) &\to nil \end{aligned}$$

Here, one could interpret a call $from(x)$ as representing an infinite list of natural numbers starting from the given argument x. Moreover, take could be interpreted as function that extracts the first n arguments of a list l where n and l are given as parameters, nil as the empty list, ":" as the list constructor and s and 0 as in Example 1.

Now the term $take(s(0), 0 : from(s(0)))$ can be simplified in two possible ways:

$$\begin{aligned} \underline{take(s(0), 0 : from(s(0)))} &\to 0 : take(0, from(s(0)))\\ &\quad\text{with rule (1.2)}\\ take(s(0), 0 : \underline{from(s(0))}) &\to take(s(0), 0 : (s(0) : from(s(s(0)))))\\ &\quad\text{with rule (1.1)} \end{aligned}$$

Obviously, the simplification using the rule (1.2) is preferable, since in one further simplification step the term $0 : nil$ is reached which is a normal form. On the other hand if one simplifies by repeatedly (and exclusively) using the rule (1.1) one cannot obtain a normal form and indeed the simplification would not terminate.

Example 2 shows that different simplification paths can lead to drastically different results. While one path quickly leads to a normal form, another path might not lead to such a normal form at all. Hence, we need criteria to (automatically) decide which rewrite rules to apply first and to what subterms. To this end so-called *reduction strategies* and *restrictions* of rewriting were developed.

1.2 Strategies and Restrictions

1.2.1 Reduction Strategies

A reduction strategy basically restricts the set of possible simplifications of a term to a subset (which is not the empty subset for reducible terms), thereby reducing or even eliminating the nondeterminism of term rewriting. Examples of such strategies for term rewriting range from simple ones, such as innermost and outermost strategies to more involved but also more powerful ones, for instance relying on various notions of sequentiality (e.g. [46, 50, 74]).

Term rewriting under an innermost strategy means that the contracted subterms do not contain redexes as proper subterms, whereas rewriting under an outermost strategy means that the contracted subterms are not contained as a proper subterm of a redex. While these strategies are easy to define, understand and implement, they often fail to serve their purpose of efficiently computing normal forms of terms.

Example 3. *Consider the following set of rewrite rules*

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

The term $g(a,a)$ has a normal form b. However, the infinite reduction $g(a,a) \to g(f(a),a) \to g(f(f(a)),a) \to \dots$ is a proper innermost as well as outermost reduction sequence. It is innermost, since the reduced subterm is always a and thus has no proper subterms. Furthermore, it is outermost, since the proper superterms of a are $g(f^n(a),a), f(a), f(f(a)), \dots$ (for arbitrary $n \in \mathbb{N}$) neither of which is a redex.

Note that in Example 3 there is an important difference in using the innermost and the outermost reduction strategy. While it is not possible to compute the normal form of $g(a,a)$ using an innermost reduction strategy, this is possible with the outermost strategy through the reduction sequence $g(a,a) \to g(a,f(a)) \to b$. In this case applying an outermost strategy still leaves some nondeterminism in the simplification of the term $g(a,a)$, since it contains two parallel redexes.

When looking at the infinite outermost reduction sequence described in Example 3, one notices that this reduction sequence does not treat all outermost redexes of $g(a,a)$, i.e. the two a's in a fair way, in the sense that the second a is never rewritten despite being an outermost redex. Indeed, if one adds a fairness constraint to the outermost strategy, yielding an *outermost-fair* reduction strategy, all normalizing terms (i.e. terms having a normal form) are rewritten to normal forms provided that the rule system is non-overlapping and left-linear (such rewrite systems are called *orthogonal*, [71]).

More advanced reduction strategies utilize the concept of needed redexes. When looking at the term $g(a, a)$ it is obvious that a reduction of the second a is necessary to compute the normal form b while the reduction of the first a is not necessary. We thus say that the second a is a *needed* redex, since its contraction is needed to compute a normal form. It was shown in [46] that every term, that is not a normal form, contains a needed redex if the rule system is orthogonal. Unfortunately, it is undecidable whether a given redex in a term is needed ([46]), so one cannot immediately derive an effective reduction strategy from this result.

However, one can effectively approximate neededness for a certain subclass of orthogonal rewrite systems called *strongly sequential* systems([50]). Then, by repeatedly contracting needed redexes a normal form is computed whenever it exists.

Example 4. *The rewrite system of Example 3 happens to be strongly sequential and in the term $g(a, a)$ the second occurrence of a can be identified as needed redex, while additionally it is detected that the first a is not needed. So when using a reduction strategy based on strong sequentiality the shortest normalizing reduction $g(a, a) \to g(a, f(a)) \to b$ is found.*

A characteristic feature of any reduction strategy in term rewriting is that every term that is not a normal form is reducible according to the strategy. In particular, this means that one easily obtains infinite rewrite derivations when applying reduction strategies naively to non-normalizing terms (i.e. terms not having a normal form).

Example 5. *Consider the rewrite rules of Example 3 and the term a. This term is not normalizing, i.e. not reducible to a normal form. Indeed we have $a \to f(a) \to f(f(a)) \to \ldots$ under every possible strategy, since in every term appearing in this infinite reduction sequence there is exactly one redex.*

So, while reduction strategies can be used to avoid some infinite reduction sequences (cf. Examples 4 and 3), others cannot be avoided (cf. Example 5). Still, oftentimes it is desirable to avoid all non-terminating reduction sequences whatsoever. One way to achieve this is the use of proper restrictions of term rewriting.

1.2.2 Restrictions of Rewriting

Restrictions of rewriting are similar to reduction strategies in that the set of possible simplifications in a term is restricted. The important difference to strategies is that this restriction may yield the empty set of possible simplifications even if the term in question is not a normal form.

While this more severe restriction of possible rewrite steps seems advantageous regarding the avoidance of infinite reduction sequences, there is a significant price to pay. This price is that when using restrictions of rewriting one has to ensure that normal forms of terms are not "missed" by too severely restricting rewriting (cf. Example 7 below). This problem hardly occurs when using reduction strategies, since there every term that is not a normal form is reducible under the strategy. Thus, whenever a rewrite computation under a strategy terminates, the unique result of the computation has been found, provided that the rewrite system is confluent (which is the case for many practical applications, e.g. for orthogonal rewrite systems).

Context-Sensitive Rewriting

Existing approaches to rewriting with restrictions mostly rely on an analysis of the syntactic structure of the term that is to be simplified. The most prominent approach, *context-sensitive rewriting* [55, 60], allows reductions only in certain arguments of certain function symbols as specified by the so-called *replacement map.*

Example 6. *Consider the rewrite system of Example 3. In order to avoid the infinite reduction sequence $a \to f(a) \to f(f(a)) \to \dots$ one could disallow the reduction of arguments of the f function symbol. This is to say that in the term $f(a)$ the only redex a may not be contracted. Note that in this case $f(a)$ is irreducible under the context-sensitive restriction despite not being a normal form. However, in this example this is not problematic, since $f(a)$ does not have a normal form, i.e. it is not normalizing.*

Indeed, when restricting rewriting w.r.t. the rules of Example 3 in the way described in Example 6 one can show that there are no infinite reduction sequences. Still, normal forms of terms can be computed if they exist. However, if we slightly modify the rewrite rules of Example 3 things get more complicated.

Example 7. *Consider the following rewrite rules, that are only slightly different from Example 3.*

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

When using the same restriction as in Example 6, i.e. disallowing reductions in the argument of f, there are still no infinite reduction sequences. However, now for instance the normal form of the term $g(a,a)$ cannot be computed, since one would have to make the reductions $g(a,a) \to g(a, f(a)) \to g(a, f(f(a))) \to b$ which is not allowed, since in the second step a is reduced which occurs in the argument of f.

Example 7 shows that the design of concrete restrictions for a given TRS, such that the restricted system has "good" properties is non-trivial. The good properties that one is usually after when using restrictions of rewriting are termination and completeness. Completeness here means completeness w.r.t. interesting derivations in the simulation of unrestricted rewriting by restricted rewriting. Obviously, there is some room of interpretation of what interesting derivations are. For instance if all derivations are interesting, then no proper restriction would be complete. However, usually one regards only those derivations as interesting that reduce a term to a normal form. In this sense the restrictions of Example 6 are complete.

In many cases demanding that all normalizing reduction sequences w.r.t. unrestricted rewriting can also be performed under a restriction is too much in the sense that if one designs a restriction satisfying this demand termination is lost. For instance in Example 2 one observes that in order to obtain a terminating restricted rewrite relation (only using context-sensitive restrictions) one has to forbid reductions in the second argument of ":". However, then the normal form of the term $\mathit{take}(s(0), 0 : (1 : \mathit{nil}))$ (which is $0 : \mathit{nil}$) cannot be computed (as one has to stop at $0 : \mathit{take}(0, 1 : \mathit{nil})$). Thus this restriction is not complete in the above sense.

One way to overcome this problem is to relax the notion of completeness. In [55, 60] an important criterion was introduced that guarantees that whenever a term is irreducible due to the restrictions but not a normal form, it is a head-normal form, i.e. no (unrestricted) reduction sequence starting from this term contains a root reduction step. The criterion says that the replacement map needs to be such that all non-variable subterms of left-hand sides of rewrite rules are allowed. Observe that this is the case for Example 3 if the argument of f is forbidden, but not in Example 7 with the same restriction, because in the left-hand side $g(x, f(f(y)))$ the non-variable subterm $f(y)$ is forbidden, since it occurs in the first argument of another f symbol. For a given rewrite system the most restrictive replacement map that fulfills the above criterion is called the *canonical replacement map*. Obviously, every replacement map that is less restrictive (we also say *more liberal*) fulfills the criterion as well.

Example 8. *Consider the rewrite rules of Example 2. By forbidding reductions in the second argument of ":" one obtains a terminating restriction of the rewrite relation. Furthermore, the corresponding replacement map (that allows reductions in all arguments of all other functions and in the first argument of ":") is more liberal than the canonical replacement map for these rewrite rules. Thus, the results computed by restricted rewriting are head-normal forms. For instance we have the derivation*

$$\mathit{take}(s(0), 0 : 1 : \mathit{nil}) \to 0 : \mathit{take}(0, 1 : \mathit{nil})$$

in the restricted system. This term is irreducible, since the term $\mathit{take}(0, 1 : \mathit{nil})$ *may not be reduced as it occurs in the second argument of a ":" symbol. However, the term is a head-normal form, which in this case is easy to see as the root symbol ":" is a constructor.*

Usually, computing head-normal forms is not satisfactory, since one actually wants to compute normal forms. However, in [60] it was shown how to obtain normal forms (whenever they exist) by repeatedly computing head-normal forms. Yet, the drawback of this approach is that if no normal forms exist the process does not terminate in general. We will see that the situation can sometimes be improved by using more complex restrictions of rewriting, in particular the *forbidden pattern* approach discussed in Section 4.2 below.

Other Restrictions

In Example 7 we saw that it was impossible to define context-sensitive replacement restrictions such that there are no infinite derivations and in addition irreducible terms w.r.t. the restricted rewrite relation have a useful connection with actual normal forms.

Hence, more complex restrictions of rewriting have been introduced that can handle more examples, in particular ones like Example 7, i.e. where a context-sensitive restriction with the canonical replacement map does not yield termination. The fundamental idea behind some of these approaches is the notion of *demand*. A redex should be allowed for reduction (by the restriction) whenever its reduction might contribute to the creation of a more outer redex (i.e. the creation of the more outer redex demands the more inner reduction step). Concrete approaches to restrictions using this basic idea can be found in [27, 57, 7]. The following example illustrates the idea of demand without going into the (quite complicated details) of concrete implementations of this idea (see also [77] for a survey and comparison of the approaches of [27, 57, 7] and others).

Example 9. *Consider the rewrite rules of Example 7 and the term $g(a,a)$. The reduction of the second a-subterm could contribute to the construction of a more outer redex, because of the rule $g(x, f(f(a))) \to b$, hence we perform a reduction*

$$g(a,a) \to g(a, f(a))$$

The term $g(a, f(a))$ is not a redex but the reduction of the second a-subterm now directly contributes to making it one. Hence, we perform this reduction yielding $g(a, f(f(a)))$. In this term further reducing any of the a subterms cannot contribute to the creation of a more outer redex, since the term itself is already a redex, hence it is safe at this point to forbid the reduction of the a-subterms and allow only the root reduction step yielding b. On the other hand when considering the term $f(a)$ the reduction of the a-subterm cannot contribute to the creation of a more outer redex and can thus safely be forbidden, in the sense that possible subsequent root reductions are not prevented by doing so.

While methods based on demand, in particular *on-demand rewriting* of [57] that properly extends the machinery of context-sensitive rewriting, are more powerful than context-sensitive rewriting, rewriting with these restrictions is significantly harder to handle than when simpler approaches are used. One problem is that already the definition of these approaches is subtle and complex (see for instance [7][Section 3.2]). Moreover, caused by the complexity of the definitions the analysis of rewrite systems with one of these complex restrictions regarding termination and completeness is hard.

In this thesis a novel approach to restrictions in term rewriting is introduced and discussed that allows for more sophisticated restrictions than context-sensitive rewriting, but is conceptually easier than other advanced restrictions.

1.3 Forbidden Patterns

From a systematic point of view we have several demands for an ideal framework of restrictions. For one its definition should be simple and intuitive. Moreover, it should be flexible enough to express the desired restrictions for many concrete rewrite systems. Finally, it should be feasible to verify properties such as termination and completeness for given rewrite systems with restrictions and the restricted rewrite relation should be "easy to handle". For instance, deciding whether a redex is allowed for reduction or not should be computationally easy and no critical nondeterminism should be artificially introduced through the restrictions (e.g. by making a confluent system non-confluent).

When defining our novel approach to restrictions of term rewriting we see the following design decisions to be made:

- What part of the context of a (sub)term is relevant to decide whether the (sub)term may be reduced or not?
- In order to specify the restricted reduction relation, is it better/advantageous to explicitly define the allowed or the forbidden part of the context-free reduction relation?
- What are the forbidden/allowed entities, for instance whole subterms, contexts, positions, etc.?
- Does it depend on the shape of the considered subterm itself (in addition to its outside context) whether it should be forbidden or not (if so, stability under substitutions may be lost)?
- Which restrictions on forbidden patterns seem appropriate (also w.r.t. practical feasibility) in order to guarantee certain desired closure and preservation properties.

In this thesis we propose a new method of restricted rewriting by using so-called *forbidden patterns*. The basic idea of rewriting with forbidden patterns is that in a term certain subterms are forbidden for reduction if they occur inside or above a specified term pattern (the *forbidden pattern*).

Example 10. *Consider the TRS from Example 3. We define a forbidden pattern restriction expressing that whenever the term (pattern) $f(x)$ matches some subterm of a term say at position p, then all positions $q > p$ (i.e. all positions below p) in this term are forbidden. Semantically, this amounts to the same restriction as in Example 6 which there was expressed through context-sensitivity.*

The advantage of forbidden pattern restrictions compared to other approaches is that the shape of the pattern terms is not restricted. Hence, one can express more fine-grained restrictions.

Example 11. *Consider the rewrite rules of Example 7. There it was argued that forbidding all reductions in arguments of f is not desirable. Hence, using a forbidden pattern $f(x)$ as in Example 10 does not improve the situation. However, if one uses a forbidden pattern $f(f(x))$ and specifies that whenever this pattern matches some subterm at position p, all subterms at positions below $p.1$ are forbidden, then the normalizing reduction sequence of Example 7 is possible within the restricted system, while the rewrite system with the restriction is terminating.*

While the formalism of forbidden pattern is quite simple in that it relies only on matching and comparison of positions within a term, the analysis of term rewriting systems restricted by forbidden pattern restrictions can be hard. Hence, it is sometimes necessary to restrict the shape of terms appearing in forbidden patterns in order to ease the verification of termination and completeness. In particular, the closure properties of rewriting are invalidated in general when using forbidden patterns.

Example 12. *Consider a rewrite rule $f(x) \to g(x)$ and a forbidden pattern $f(f(a))$ forbidding reductions at position $p.1$ whenever $f(f(a))$ matches at position p. We have $f(f(x)) \to f(g(x))$ but $f(f(x))\sigma = f(f(a)) \not\to f(g(a)) = f(g(x))\sigma$, hence closure under substitutions is violated. Moreover, $f(a) \to g(a)$ but $C[f(a)] = f(f(a)) \not\to f(g(a)) = C[g(a)]$, hence closure under contexts is violated as well.*

The main results of Section 4 below concern the analysis of rewrite systems (actually the verification of certain properties of these systems) when only a restricted class of forbidden patterns is used. One of the most important research tasks is to identify classes of forbidden patterns that are sufficiently large to express the desired restrictions for many rewrite systems, but on the other hand sufficiently small so that efficient methods for the analysis of these systems can be defined. An easy example of a relevant class of forbidden patterns are those where only linear terms (i.e. patterns) are used. More sophisticated classes of forbidden patterns are analyzed in Section 4.

The goal is that within these classes of forbidden patterns termination analysis and completeness analysis for concrete rewrite systems and forbidden patterns is feasible. The most important class that we discuss in this thesis are so-called *canonical* forbidden patterns. When using canonical forbidden patterns to restrict a given rewrite system, terms that are irreducible w.r.t. to the restricted rewrite system are head-normal forms w.r.t. the underlying unrestricted one (similar to the situation when a canonical replacement map is used in context-sensitive rewriting).

For this class of patterns we propose two approaches to verify termination. The first one uses a transformation of a restricted rewrite system into an unrestricted one such that (ground) termination of both systems coincides. The other approach (which is practically more powerful according to our evaluation) is based on an extension of the well-known dependency pair framework that has been used successfully in the termination analysis of unrestricted rewriting (cf. e.g. [33]), context-sensitive rewriting (cf. e.g. [2]) and other extended notions of rewriting (cf. e.g. [62, 6]).

1.4 Conditional Rewriting

A practically important extension of the formalism of term rewriting is conditional term rewriting. There, conditions are attached to rewrite rules restricting the applicability of the rules. That is to say, that a rewrite rule may only be used for a simplification step if the corresponding conditions are satisfied. The conditions in turn are parametrized by the concrete instantiation of the rule that is to be applied.

Example 13. *Consider the following conditional rewrite system.*

$$\begin{aligned}
insert(x, y : ys) &\rightarrow x : y : ys \Leftarrow x \leq y \rightarrow^* true && (1.3)\\
insert(x, y : ys) &\rightarrow y : insert(x, ys) \Leftarrow x \leq y \rightarrow^* false && (1.4)\\
insert(x, nil) &\rightarrow x : nil\\
s(x) \leq 0 &\rightarrow false\\
0 \leq x &\rightarrow true\\
s(x) \leq s(y) &\rightarrow x \leq y
\end{aligned}$$

If interpreted accordingly, these rules model the insertion of an element (a number) into a sorted list. For instance we have

$$\begin{aligned}
insert(s(0), 0 : s(s(0)) : nil) &\rightarrow 0 : insert(s(0), s(s(0)))\\
&\rightarrow 0 : s(0) : s(s(0)) : nil
\end{aligned}$$

In the first step of this reduction sequence the conditional rule (1.4) is applied. In particular, the instance of the rule obtained by the substitution σ where $x\sigma = s(0), y\sigma = 0$ and $ys\,\sigma = s(s(0)) : nil$ is used. This rule may only be applied if the corresponding condition is satisfied, i.e. if $(x \leq y)\,\sigma \rightarrow^* false\ \sigma$ holds. Obviously, we have $(x \leq y)\,\sigma = s(0) \leq 0 \rightarrow false = false\ \sigma$. Hence, the rule may be applied when this particular substitution is used. Analogously, in the second step of our reduction sequence the conditional rule (1.3) is used to obtain the normal form $0 : s(0) : s(s(0)) : nil$.

It is not surprising that checking a concrete condition can be quite hard. Indeed, the one-step conditional rewrite relation is in general not decidable ([12, 48]). Hence, the tasks of implementing the use of conditional rewrite rules in rewrite engines and of analyzing conditional rewrite systems for properties such as termination and confluence are highly non-trivial.

Traditionally, transformations have been used for both tasks. For the use of conditional rewrite rules in rewrite engines the conditional system is transformed into an unconditional one, such that whenever a term s is reachable from a term t though conditional rewriting it should also be reachable through ordinary rewriting in the transformed system. This property of a transformation is called *completeness w.r.t. simulations* or *simulation-completeness* (beware that in [69] the notion of simulation-completeness has a different meaning). Several concrete transformations have been suggested that have the property of completeness w.r.t. simulations, e.g. [64, 63, 85, 83, 34]. It turns out that the dual property of *soundness w.r.t. simulations* or *simulation-soundness*, meaning that a term s is reachable from a term t in the conditional system if s is reachable from t in the transformed system, is much harder to obtain when designing a transformation. In fact, none of the cited transformations is in general simulation-sound in this sense.

For the analysis of termination usually some of the above mentioned transformations are used. Obviously, completeness w.r.t. simulation, which all these transformations enjoy, is sufficient to derive the absence of infinite conditional derivations whenever the transformed unconditional system is terminating. However, there are two problems with this approach. First, due to the lack of soundness w.r.t. transformations the termination criterion is only sufficient but not a full characterization of conditional termination by unconditional termination. Second, and more severely, infinite reduction chains are not the only source of non-termination in conditional rewriting. It was already mentioned that even the one-step rewrite relation is not decidable in general for conditional rewriting. Thus, the attempt to evaluate a single condition which is not explicitly part of a conditional rewrite sequence might already not terminate.

This problem was dealt with by introducing the notion of *effective termination* ([10]). A conditional rewrite system is effectively terminating if there are no infinite derivations and in addition the one-step rewrite relation is decidable. In [72, 73] it was shown that for the unraveling transformation of [72] which is a modified version of the transformation of [64] that is applicable to a wider class of conditional systems, termination of the transformed system implies effective termination of the original conditional system.

Yet, in [61] it was argued that in practical applications effective termination might still not be sufficient to guarantee finiteness of all computations w.r.t. to a conditional rewrite system. This is illustrated by the following example.

Example 14 ([61, 73])**.** *Consider the following rewrite system $\mathcal{R}$.*

$$a \;\to\; b \Leftarrow f(a) \to b$$

$\mathcal{R}$ is effectively terminating, which is trivial, because the rewrite relation is empty altogether. However, the standard approach to evaluate the term a is to attempt to recursively evaluate $f(a)$, i.e. the left-hand side of the condition of the rule defining a. However, this is problematic, since when evaluating $f(a)$ one might again try to apply the only rewrite rule of $\mathcal{R}$ ending up checking the condition again. Hence, naively pursuing this eager checking of conditions is another source of infinity in conditional rewrite systems.

In order to capture conditional rewrite systems where also this source of infinity does not occur, the notion of *operational termination* of conditional rewrite systems was introduced in [61] (and in a more general form for *membership equational programs* in [18]). Roughly, a conditional rewrite system (resp. membership equational program) is operationally terminating if there are no infinite proof trees in a certain logical inference system modeling the recursive evaluation of conditions.

Fortunately, it turns out that, as for effective termination, termination of the unconditional rewrite system obtained from a conditional one through the transformation of [72] implies operational termination of the conditional TRS (cf. [61] and [72]).

In Chapter 3 of this thesis we use a slightly modified version of this transformation to obtain an actual characterization of operational termination for so called deterministic conditional rewrite systems (DCTRSs) which is arguably the most general class of conditional rewrite systems for which operational termination is interesting. This is because CTRSs that are not deterministic contain extra variables, either in the right-hand sides of their rules or the left-hand sides of conditions, which can be arbitrarily instantiated during rewriting computations. This possibility of arbitrary instantiations leads to non-operational termination in all but pathological cases. We achieve the characterization of operational termination by imposing a restriction on unconditional rewrite systems obtained by our transformation using context-sensitivity and weaken the notion of termination to local termination (c.f. e.g. [26]). By using this approach to characterize operational termination of a DCTRS by a local termination property of a corresponding unconditional (context-sensitive) TRS we not only obtain a powerful method to verify operational termination, but also to disprove operational termination.

1.5 Contributions

In Chapter 3 we introduce the notion of context-sensitive quasi reductivity (Definition 1) and prove that this property of DCTRSs is equivalent to operational termination for this class of conditional TRSs (Corollary 4). Moreover, we introduce a version U_{cs} of the unraveling transformation of [72] where the resulting unconditional TRS is restricted by context-sensitive replacement restrictions (Definition 4). We prove that this transformation is simulation-sound and simulation-complete (Theorems 2 and 1) and can thus be used to verify operational termination of DCTRS by (context-sensitive) termination of the obtained unconditional TRSs. Yet, context-sensitive termination of a transformed TRS $U_{cs}(\mathcal{R})$ and operational termination of the initial DCTRS $\mathcal{R}$ do not coincide in general (Example 21).

However, we prove that operational termination of a DCTRS $\mathcal{R}$ does coincide with local termination of $U_{cs}(\mathcal{R})$ where local termination here means that one is only interested in reduction sequences starting from terms over the original signature of $\mathcal{R}$ (Corollary 4).

Furthermore, if one considers the more general property of $C_{\mathcal{E}}$-(operational) termination instead of operational termination, it turns out that this property is indeed completely characterized by $C_{\mathcal{E}}$-termination of the context-sensitive TRS obtained by our transformation (Theorem 6). By $C_{\mathcal{E}}$-(operational) termination of a (DC)TRS $\mathcal{R}$ we mean (operational) termination of the (DC)TRS $\mathcal{R}'$ obtained by disjointly adding the two rules $C(x, y) \to x, C(x, y) \to y$ to $\mathcal{R}$ (where C is a new function symbol).

Finally, we also propose an approach to the automated verification of local termination in the above sense for CSRSs obtained by our transformation. This approach is based on the well-known dependency pair framework of [9, 33] and implemented in the termination tool VMTL [82]. Benchmark tests showed that the approach succeeds in proving local termination of rewrite systems where global termination does not hold.

The results of this chapter were published in [78] (Sections 3.2.1 and 3.2.2), [80] (Sections 3.2.1, 3.2.2 and 3.2.3) and [81] (the whole Chapter 3).

In Chapter 4 we introduce a novel approach to restrictions in term rewriting. This approach is based on the notion of forbidden patterns. We discuss several design decisions and provide criteria for confluence (Theorems 12 and 13), completeness (Theorem 20) and termination. Regarding termination we introduce one method based on a transformation (Definition 28 and Theorem 14) and one method that utilizes a contextual extension of the dependency pair framework of [9, 33] (Section 4.4.2).

We also introduce a method for the automated synthesis of suitable forbidden patterns for a given rewrite system, which is based on our contextual extension of the dependency pair framework (Section 4.6).

Various examples are provided where the use of existing restrictions and/or strategies would not yield the desired properties of rewriting while the use of forbidden patterns does (e.g. Examples 32 and 34).

Moreover, since our approach is more general than many others, other restrictions and strategies such as context-sensitive rewriting and innermost/outermost strategies can easily be expressed by using forbidden patterns. Thus, our criteria for completeness and termination can be reused also for these simpler approaches. We will exemplify this idea by using our contextual dependency pair framework to prove outermost termination of TRSs (cf. Section 4.4.2).

The results of this chapter were published in [37] (Sections 4.2.1, 4.2.2, 4.4.1 and 4.5), [39] (Sections 4.4.2, 4.4.2.1 and 4.6) and [38] (giving an overview of Section 4.4.2.2).

Finally, in Chapter 5 we introduce the termination laboratory VMTL which contains implementations of all the methods for termination analysis introduced in Chapters 3 and 4 and which was used for all benchmarks presented in this thesis. Moreover, VMTL provides its user with an open and extensible implementation of the dependency pair framework and particularly allows the user to define custom strategies for proof search and to add dependency pair processors in a modular way. A system description of VMTL was published in [82].

Summarizing we think that the main and most interesting contributions of this thesis are:

- The characterization of operational termination of deterministic DCTRSs $\mathcal{R}$ by a local termination property of the context-sensitive TRS $U_{cs}(\mathcal{R})$ and in addition the proofs of simulation-soundness and simulation-completeness of the transformation U_{cs} as well as the characterization of $C_{\mathcal{E}}$-operational termination of DCTRSs $\mathcal{R}$ by $C_{\mathcal{E}}$-termination of the CSRS $U_{cs}(\mathcal{R})$.

- The introduction of rewriting with forbidden patterns; in particular, criteria for confluence, completeness and termination for rewriting under forbidden pattern restrictions.
- The development of the termination laboratory VMTL, that has been used as a test bench for performance measurements of the theoretic results concerning termination of this thesis and contains implementations of many state-of-the-art termination methods for ordinary, context-sensitive and conditional TRSs.

1.6 Outline

In Chapter 2 we present all notions and notations used in the subsequent chapters. Chapter 3 is concerned with conditional rewrite systems and the verification of operational termination. The main results of this chapter are Theorems 2, 1, 4, 5, 6, 8, 9, 10 and 11. In Chapter 4 we introduce rewriting with forbidden patterns. The main results of this section are Theorems 12 and 13 regarding confluence, Theorems 14, 15, 16 and 19 regarding termination and Theorem 20 regarding completeness of rewriting with forbidden patterns. Chapter 5 contains a description of the termination tool VMTL which provides implementations of the termination methods introduced in Chapters 3 and 4. Chapter 6 contains a summary and discussion of the results in this thesis as well as several pointers to related work.

Chapter 2

Preliminaries

2.1 Abstract Reduction Systems

An *abstract reduction system* is a pair $(A, \to)$ where A is some set and $\to$ is a binary relation on A. Elements $a \in A$ for which no element $b \in A$ with $a \to b$ exists are called *normal forms*. The set of all normal forms of an abstract reduction system $\mathcal{A}$ is denoted by $NF(\mathcal{A})$ ($NF(\mathcal{A}) \subseteq A$)). An element is *reducible* if it is not a normal form. We denote by $\to^+$ ($\to^{0/1}$, $\to^*$) the transitive (reflexive, reflexive-transitive) closure of the relation $\to$. Moreover, by $\leftarrow$ we denote the inverse relation of $\to$ and by $\leftrightarrow$ the union of $\to$ and $\leftarrow$. Joinability of two elements a and b is denoted by $a \downarrow b$, i.e. $a \downarrow b \Leftrightarrow \exists c.a \to^* c \leftarrow^* b$. An element b is a normal form of a if $a \to^* b$ and b is a normal form.

The relation $\to$ is *well-founded* (*terminating* or *strongly normalizing*) if there is no infinite sequence $a_1, a_2, \ldots$ of elements of A such that $a_i \to a_{i+1}$ holds for all $i \geq 1$. We write $SN(\to)$. We call (possibly infinite) sequences of elements $a_1, \ldots, a_i \ldots$ where $a_i \to a_{i+1}$ holds for all $i \geq 1$ reduction sequences or reduction chains. Furthermore, $\to$ is confluent (or equivalently it has the *Church-Rosser* property) if $b \leftarrow^* a \to^* c$ implies $b \downarrow c$. We write $CR(\to)$. It has the diamond property if $b \leftarrow a \to c$ implies that there exists an element d such that $b \to d \leftarrow c$. An abstract reduction system is *weakly normalizing* ($WN(\to)$) if each element a of A has a normal form.

We say that an element $a \in A$ is strongly (weakly) normalizing ($SN(a)$ resp. $WN(a)$) if there is no infinite reduction sequence starting from a (resp. if there exists a normal form of a).

2.2 Term Rewriting

A *signature* is a set of function symbols with arities. In this thesis the arities of function symbols are mostly left implicit, hence we say for instance that a function symbol f is part of some signature without explicit reference to the arity of f. In case an explicit reference to the arity of f is needed we write $ar(f)$. Function symbols with arity 0 are called *constants*. Signatures are denoted by $\mathcal{F}$, $\mathcal{F}'$, $\mathcal{F}_1$ etc. By Var (or just V) we denote a countably infinite set of variables which we write as $x, y, z, x_1, y_1, z_1, x', y', z', \ldots$.

A *term* over a signature $\mathcal{F}$ is a tree whose nodes are labelled by elements from $\mathcal{F} \cup V$ where only leave nodes are labelled by variables and the number of children of each inner node labelled by $f \in \mathcal{F}$ is equal to the arity of f. The root label of a term s is denoted as $root(s)$. Terms over a signature $\mathcal{F}$ are denoted by $\mathcal{T}(\mathcal{F}, V)$. Terms not containing variables are called *ground* terms and the set of such terms is denoted by $\mathcal{T}(\mathcal{F}, \emptyset)$ (or simply $\mathcal{T}(\mathcal{F})$).

A many-sorted signature $\mathcal{F}$ is a pair (S, Ω) where S is a set of *sorts* and Ω is a family of (mutually disjoint) sets of typed function symbols: $\Omega(\Omega)_{\omega,s}$ where $\omega \in S^*$ and $s \in S$. We also say, f is of type $\omega \to s$ (or just s if $\omega = \epsilon$) if $f \in \Omega_{\omega,s}$. $V = (V_s \mid s \in S)$ is a family of (mutually disjoint) countably infinite sets of typed variables (with $V \cap \Omega = \emptyset$). The set $\mathcal{T}(\mathcal{F}, V)_s$ of (well-formed) terms of sort s is the least set containing V_s, and whenever $f \in \Omega_{(s_1,\ldots,s_n),s}$ and $t_i \in \mathcal{T}(\mathcal{F}, V)_{s_i}$ for all $1 \leq i \leq n$, then $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, V)_s$. The sort of a term t is denoted by $sort(t)$.

Positions in terms are sequences of natural numbers (written for example as $n_1.n_2.n_3$). The empty sequence is denoted by ϵ. The set of positions $Pos(s)$ of a term s is recursively defined by $Pos(s) = \{\epsilon\}$ if s is a variable or a constant, and $Pos(s) = \{i.p \mid 1 \leq i \leq ar(f), p \in Pos(s|_i)\}$ if $root(s) = f$ and $ar(f) \geq 1$. Here $s|_i$ denotes the subtree rooted by the i^{th} child node of the root node of s. More generally $s|_p$ for some position $p \in Pos(s)$ denotes the subterm at position p of s and is defined by $s|_\epsilon = s$ and $s|_{i.p'} = (s|_i)|_{p'}$. We use a partial order $\leq$ on positions defined by $p \leq q$ if p is a (not necessarily proper) prefix of q. Positions p and q are parallel, denoted $p||q$ if neither $p \leq q$, nor $q \leq p$. By $s[t]_p$ (for $p \in Pos(s)$) we denote the term where the subterm of s at position p has been replaced by t. Formally, $s[t]_\epsilon = t$, $s[t]_i = f(s_1, \ldots, s_{i-1}, t, s_{i+1}, \ldots, s_{ar(f)})$ if $s = f(s_1, \ldots, s_{ar(f)})$, $i \in \mathbb{N}$ and $1 \leq i \leq ar(f)$ and $s[t]_{i.p'} = s[s|_i[t]_{p'}]_i$ otherwise. The size $|p|$ of a position p is given by $|\epsilon| = 0$ and $|i.p'| = 1 + |p'|$. The depth $|t|$ of a term t is given by $max_{p \in Pos(t)}(|p|)$.

$Var(s)$ denotes the set of variables of a term s, i.e. $Var(s) = \emptyset$ if s is a constant, $Var(s) = \{s\}$ if s is a variable and $Var(s) = \bigcup_{1 \leq i \leq n} Var(s_i)$ if $s = f(s_1, \ldots, s_n)$. Slightly abusing notation we also write $Var(S)$ for a set S of terms which is defined by $\bigcup_{s \in S} Var(s)$. The set of variable positions $Pos_V(s)$ of a term s are given by $\{p \in Pos(s) \mid s|_p \in Var(s)\}$. The non-variable positions $Pos_{\mathcal{F}}(s)$ of s are $Pos(s) \setminus Pos_V(s)$. A term s is called linear if each variable occurs at most once in s.

Given a signature $\mathcal{F}$ a *substitution* is a mapping from variables to terms over $\mathcal{F}$, i.e. $Var \to \mathcal{T}(\mathcal{F}, V)$. Substitutions are denoted by $\sigma, \theta, \tau, \ldots$. Substitutions can be extended to mappings from terms to terms in a straightforward way, i.e. $\sigma(s) = \sigma(x)$ if $s = x \in Var$ and $\sigma(s) = f(\sigma(s_1), \ldots, \sigma(s_n))$ if $s = f(s_1, \ldots, s_n)$. The domain of a substitution σ is denoted by $Dom(\sigma)$. The codomain is denoted by $Codomain(\sigma)$ and defined by $\{s \in \mathcal{T}(\mathcal{F}, V) \mid \exists x \in Dom(\sigma). x\sigma = s\}$. For the sake of readability we use postfix notation for substitutions, i.e. we write $s\sigma$ instead of $\sigma(s)$. A term s matches a term t, written $s \leq_{mr} t$ if there exists a substitution σ such that $s\sigma = t$. We also say that t is an instance of s. The induced matching relation $\leq_{mr}$ is a quasi-order (i.e. a reflexive and transitive binary relation) on terms. A substitution σ is said to be more general than another substitution σ' if $Dom(\sigma) = Dom(\sigma')$ and $x\sigma \leq_{mr} x\sigma'$ for all $x \in Dom(\sigma)$. Two terms s and t unify if there exists a substitution σ such that $s\sigma = t\sigma$. Among all substitutions σ for which $s\sigma = t\sigma$, the most general one is unique up to renaming of the variables occurring in terms of $Codomain(\sigma)$ and is called the *most general unifier* (mgu).

A *term rewrite rule* is an ordered pair of terms l, r denoted by $l \to r$ where $Var(r) \subseteq Var(l)$ and $l \notin Var$. A *term rewrite system* (TRS) $\mathcal{R}$ is a pair $(\mathcal{F}, R)$ where $\mathcal{F}$ is a signature and R is a set of rewrite rules consisting of terms over $\mathcal{F}$ (abusing notation we sometimes identify a rewrite system $\mathcal{R}$ with the set of its rewrite rules, for instance in the statement $l \to r \in \mathcal{R}$ where $l \to r$ is a rewrite rule and $\mathcal{R}$ is a TRS). Sorted rewrite rules are rewrite rules $l \to r$ where $sort(l) = sort(r)$. Types of terms and rewrite rules are made explicit only if they are used and relevant. A TRS is called left-linear (right-linear) if the left-hand sides (right-hand sides) of all rewrite rules are linear. Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS and $s \in \mathcal{T}(\mathcal{F}, V)$. Moreover, let $l \to r \in R$ be a rewrite rule such that l matches $s|_p$ for some $p \in Pos_{\mathcal{F}}(s)$ (we sometimes say that the rule $l \to r$ matches $s|_p$). Then s rewrites to $t = s[r\sigma]_p$ at position p using the rule $l \to r$. We write $s \overset{p}{\to}_{l\to r,\mathcal{R}} t$, $s \overset{p}{\to}_{\mathcal{R}} t$, $s \to_{\mathcal{R}} t$ or just $s \to t$ if the used rewrite rule, the position or the TRS are clear from the context or of no relevance. The parallel rewrite relation w.r.t. $\mathcal{R}$ is given by $s \mathrel{+\!\!\!\!\!\!\!\;\longrightarrow}_{\mathcal{R}} t$ if and only if $s \overset{p_1}{\to}_{\mathcal{R}} s_2 \overset{p_2}{\to}_{\mathcal{R}} \ldots \overset{p_{n-1}}{\to}_{\mathcal{R}} s_n \overset{p_n}{\to}_{\mathcal{R}} t$ and the positions $p_1, \ldots, p_n$ are pairwise parallel. Moreover, if l unifies with $s|_p$ (with mgu θ) for some $p \in Pos_{\mathcal{F}}(s)$, then s *narrows* to $s[r]_p\theta = t$, we write $s \overset{p}{\leadsto}_{l\to r,\mathcal{R}} t$.

In a reduction $s \overset{p}{\to}_{l\to r,\mathcal{R}} t$, the term $s|_p = l\sigma$ (for some substitution σ) is called a *redex*. Slightly abusing this notion, given a term s and a TRS $\mathcal{R}$ we say that all subterms of s at position q where $s|_q = l\sigma$ for some $l \to r \in \mathcal{R}$ and some substitution σ are redexes (q is called the redex position).

Furthermore, let $s_1 \overset{p_1}{\leftarrow}_{l_1\to r_1,\mathcal{R}} s \overset{p_2}{\to}_{l_2\to r_2,\mathcal{R}} s_2$ be a divergence such that $p_1 \leq p_2$. We say this divergence is due to a *variable overlap* if $p_2 \geq p_1.q$ for some position $q \in Pos_V(l_1)$. On the other hand it is a *critical overlap* if $p_2 = p_1.q$ for some position $q \in Pos_{\mathcal{F}}(l_1)$. In that case the pair of terms (s_1, s_2) is called a *critical pair*. Slightly abusing terminology we say that a rewrite rule $l \to r$ overlaps a term t if l and $t|_p$ unify for some position $p \in Pos_{\mathcal{F}}(t)$.

2.3 Context-Sensitive Term Rewriting

A *replacement map* μ (for a signature $\mathcal{F}$) is a mapping $\mathcal{F} \to \mathcal{P}(\mathbb{N})$ from symbols of $\mathcal{F}$ to sets of natural numbers such that $\mu(f) \subseteq \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}$. Given a term s, the *replacing* (or *allowed*, *active*) positions $Pos^{\mu}(s) \subseteq Pos(s)$ of

s w.r.t. a replacement map μ are given by $\{\epsilon\}$ if s is a constant or a variable and $\{\epsilon\} \cup \{i.p \mid i \in \mu(f), p \in Pos^{\mu}(s_i)\}$ if $s = f(s_1, \ldots, s_n)$. We denote the replacing non-variable positions of a term s by $Pos^{\mu}_{\mathcal{F}}(s)$ and the replacing variable positions of s by $Pos^{\mu}_{V}(s)$. The non-replacing positions of a term s w.r.t. a replacement map μ are denoted by $Pos^{\overline{\mu}}(s)$. The set of variables of a term t that occur at replacing positions is denoted by $Var^{\mu}(t)$. The set of variables that occur at non-replacing positions in t are denoted by $\overline{Var^{\mu}}(t)$ We call a replacement map μ *more restrictive* than a replacement map μ' (both for the same signature $\mathcal{F}$) if $\mu(f) \subseteq \mu'(f)$ for all $f \in \mathcal{F}$. This relation is a partial order on replacement maps ([55, 60]). An ordering $\succ$ on terms $\mathcal{T}(\mathcal{F}, V)$ is called μ-monotonic if f is monotonic in its i^{th} argument whenever $i \in \mu(f)$ for all $f \in \mathcal{F}$, i.e.,

$$s_i \succ t_i \Rightarrow f(s_1, \ldots, s_{i-1}, s_i, s_{i+1} \ldots, s_n) \succ f(s_1, \ldots, s_{i-1}, t_i, s_{i+1} \ldots, s_n).$$

A TRS with replacement map μ is called a *context-sensitive rewrite system* (CSRS) and denoted as pair $(\mathcal{R}, \mu)$. Context-sensitive rewriting w.r.t. a CSRS $(\mathcal{R}, \mu)$ (denoted by $\rightarrow_{\mu,\mathcal{R}}$) is rewriting at replacing positions, i.e. $s \xrightarrow{p}_{\mu,\mathcal{R}} t \Leftrightarrow s \xrightarrow{p}_{\mathcal{R}} t$ and $p \in Pos^{\mu}(s)$.

Given a CSRS $(\mathcal{R} = (\mathcal{F}, R), \mu)$, the relation of context-sensitive narrowing (written $\rightsquigarrow^{\mu}_{\mathcal{R}}$) is defined as $t \rightsquigarrow^{\mu}_{\mathcal{R}} s$ if there is a replacing non-variable position p in t such that $t|_p$ and l unify ($l \rightarrow r \in R$ and we assume that t and $l \rightarrow r$ do not share any variables) with mgu θ and $s = t[r]_p\theta$. We say that s is a *one-step, context-sensitive narrowing* of t. Note that in contrast to ordinary rewriting, here we allow rules in R to have extra variables in the right-hand sides and variable left-hand sides. The reason for this general definition of narrowing is that we are going to use a *backward narrowing* relation that is induced by reversing all rules of a TRS (cf. Definition 13 below).

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$ the *canonical* replacement map is the most restrictive replacement map μ such that for every $l \rightarrow r \in R$ $Pos_{\mathcal{F}}(l) \subseteq Pos^{\mu}(l)$. By canonical context-sensitive rewriting we mean context-sensitive rewriting in the presence of the canonical or a less restrictive replacement map.

Conditional Rewriting

A conditional term rewriting system (CTRS) $\mathcal{R}$ (over some signature $\mathcal{F}$) consists of rules $l \rightarrow r \Leftarrow c$ where c is a conjunction of equations $s_i = t_i$. Equality in the conditions may be interpreted (recursively) e.g. as $\leftrightarrow^*$ (semi-equational case), as $\downarrow$ (join case), or as $\rightarrow^*$ (oriented case). In the latter case, if all right-hand sides of conditions are ground terms that are irreducible w.r.t. the unconditional version $\mathcal{R}_u = \{l \rightarrow r \mid l \rightarrow r \Leftarrow c \in \mathcal{R}\}$ of $\mathcal{R}$, the system is said to be a *normal* one.

According to the distribution of variables, a conditional rule $l \rightarrow r \Leftarrow c$ may satisfy (1) $Var(r) \cup Var(c) \subseteq Vars(l)$, (2) $Var(r) \subseteq Vars(l)$, (3) $Vars(r) \subseteq Vars(l) \cup Vars(c)$, or (4) no variable constraints. If all rules of a CTRS R are of type (i), $1 \leq i \leq 4$, we say that R is an i-CTRS. Given a conditional rewrite rule

$l \to r \Leftarrow c$ and a variable x such that $x \in Var(r) \cup Var(c)$ but $x \notin Var(l)$, we say that x is an *extra variable.*

In this thesis we are mainly concerned with oriented CTRSs and in particular with so-called *deterministic* CTRSs. A deterministic CTRS (DCTRS) is an oriented 3-CTRS where for each rule $l \to r \Leftarrow s_1 \to t_1, \ldots, s_n \to t_n$ it holds that $Var(s_i) \subseteq Var(l) \cup \bigcup_{j=1}^{i-1} Var(t_j)$. The rewrite relation of an oriented CTRS $\mathcal{R}$ is inductively defined as follows $R_0 = \emptyset$, $R_{j+1} = \{l\sigma \to r\sigma \mid l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n \in \mathcal{R} \wedge s_i\sigma \to^*_{R_j} t_i\sigma \text{ for all } 1 \leq i \leq n\}$, and $\to_{\mathcal{R}} = \bigcup_{j \geq 0} \to_{R_j}$.

A DCTRS $(\mathcal{F}, R)$ is called *quasi-reductive*, cf. [73], [29], if there exists an extension $\mathcal{F}'$ of $\mathcal{F}$ and a well-founded partial order $\succ$ on $\mathcal{T}(\mathcal{F}', V)$, which is monotonic, i.e., closed under contexts, such that for every rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n \in R$, every $\sigma \colon V \to \mathcal{T}(\mathcal{F}', V)$ and every $i \in \{0, \ldots, n-1\}$:

- If $s_j\sigma \succeq t_j\sigma$ for every $1 \leq j \leq i$, then $l\sigma \succ_{st} s_{i+1}\sigma$.
- If $s_j\sigma \succeq t_j\sigma$ for every $1 \leq j \leq n$, then $l\sigma \succ r\sigma$.

Here $\succ_{st} = (\succ \cup \rhd)^+$ ($\rhd$ denotes the *proper subterm* relation).

A DCTRS $\mathcal{R} = (\mathcal{F}, R)$ is *quasi-decreasing* [73] if there is a well-founded partial ordering $\succ$ on $\mathcal{T}(\mathcal{F}, V)$, such that $\to_{\mathcal{R}} \subseteq \succ$, $\succ = \succ_{st}$, and for every rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n \in R$, every substitution σ and every $i \in \{0, \ldots, n-1\}$ it holds that $s_j\sigma \to^*_{\mathcal{R}} t_j\sigma$ for all $j \in \{1, \ldots, i\}$ implies $l\sigma \succ s_{i+1}\sigma$.

In Section 3 we are concerned with the property of operational termination of DCTRS. In [61, 18] *operational termination* of (D)CTRSs is defined via the absence of infinite well-formed trees in a certain logical inference system. In the case of DCTRSs, this notion is shown to be equivalent to quasi-decreasingness [61].

The latter notions are related as follows ([73], [61]):

$$\text{quasi-reductivity} \Rightarrow \text{quasi-decreasingness} \Leftrightarrow \text{operational termination}$$

2.4 The (Context-Sensitive) Dependency Pair Framework

Given a TRS $\mathcal{R} = (\mathcal{F}, R)$, the signature $\mathcal{F}$ is partitioned into its *defined* and *constructor* symbols $\mathcal{D} \uplus \mathcal{C}$, where the defined symbols are exactly those that occur as root symbols of the left-hand sides of rules in R. A term t is *hidden* w.r.t. a CSRS $(\mathcal{R} = ((\mathcal{D} \uplus \mathcal{C}, R), \mu))$ if $root(t) \in \mathcal{D}$ and t appears non-μ-replacing in the right-hand side of a rule of $\mathcal{R}$. Moreover, we say that a function f *hides* a position i if there is a rule $l \to r \in R$ such that some term $f(r_1, \ldots, r_i, \ldots, r_n)$ occurs at a non-replacing position of r, $i \in \mu(f)$ and r_i contains a (not necessarily proper) subterm $r_i|_p$, such that $p \in Pos^{\mu}(r_i)$ and either $r_i|_p \in Var$ or $root(r_i|_p) \in \mathcal{D}$. By $t^\sharp$ we denote the term $f^\sharp(t_1, \ldots, t_n)$, where $t = f(t_1, \ldots, t_n)$ and $f^\sharp$ is a new *dependency pair symbol.*

The set of context-sensitive dependency pairs ([2]) of a CSRS $(\mathcal{R}, \mu)$, denoted $DP(\mathcal{R}, \mu)$, is $DP_o(\mathcal{R}, \mu) \cup DP_u(\mathcal{R}, \mu)$ where the set of "ordinary" DPs $DP_o(\mathcal{R}, \mu)$ is given by

$$\{l^\sharp \to s^\sharp \mid l \to r \in R, r \trianglerighteq_\mu s, root(s) \in \mathcal{D}, l \not\triangleright_\mu s\}$$

and $DP_u(\mathcal{R}, \mu)$ is the union of the following "unhiding" dependency pairs:

- $\{l^\sharp \to D^\sharp(x) \mid l \to r \in R, x \in Var^\mu(r) \setminus Var^\mu(l)\}$,
- $D^\sharp(f(x_1, \ldots, x_i, \ldots, x_n)) \to D^\sharp(x_i)$ for every function symbol f of any arity n and every $1 \leq i \leq n$ where f hides position i, and
- $D^\sharp(t) \to t^\sharp$ for every hidden term t.

Here, $D^\sharp$ is a fresh auxiliary function symbol that is used to avoid collapsing dependency pairs (i.e. DPs whose right-hand side is a variable; cf. [2] for further details). The relation $\trianglerighteq_\mu$ is defined as $s \trianglerighteq_\mu t$ if $s = s[t]_p$ and $p \in Pos^\mu(t)$.

We denote by $\mathcal{F}^\sharp$ the signature $\mathcal{F}$ plus all dependency pair symbols plus the new symbol $D^\sharp$. The replacement map μ is extended into $\mu^\sharp$ where $\mu^\sharp(f) = \mu(f)$ if $f \in \mathcal{F}$, $\mu^\sharp(f^\sharp) = \mu(f)$ if $f^\sharp$ is a dependency pair symbol and $\mu(D^\sharp) = \emptyset$.

Let DP and $\mathcal{R}$ be TRSs and μ be a replacement map for their combined signature. A (possibly infinite) sequence of rules $s_1 \to t_1, s_2 \to t_2, \ldots$ (having disjoint sets of variables) from DP is a $(DP, \mathcal{R}, \mu)$-chain if there is a substitution σ, such that $t_i\sigma \to^*_{\mathcal{R},\mu} s_{i+1}\sigma$ for all $i > 0$. We say that σ *enables* the chain $s_1 \to t_1, s_2 \to t_2, \ldots$

We call a triple $(DP, \mathcal{R}, \mu)$, where DP and $\mathcal{R}$ are TRSs and μ is a replacement map for the combined signatures of DP and $\mathcal{R}$, a *(context-sensitive) dependency pair problem* (CS-DP problem). A context-sensitive dependency pair problem is finite if there is no infinite $(DP, \mathcal{R}, \mu)$-chain and infinite otherwise.

A CSRS $(\mathcal{R}, \mu)$ is μ-terminating if and only if the dependency pair problem $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu)$ is finite ([2]).

Chapter 3

Using Context-Sensitivity in Unravelings of Conditional TRSs

3.1 Introduction

When analyzing the termination behaviour of conditional TRSs, it turns out that the proof-theoretic notion of *operational termination* [61, 18] is more adequate than ordinary termination in the sense that practical evaluations w.r.t. operationally terminating DCTRSs always terminate (which is indeed not true for other similar notions like *effective termination* [73], cf. Example 14).

For the analysis of operational termination of DCTRSs the equivalent property of quasi-decreasingness is usually used [61]. However, to the author's knowledge, there are no automated techniques to verify quasi-decreasingness of DCTRSs directly (intuitively, the reason why it is hard to develop such techniques is that the definition of quasi-decreasingness relies on the rewrite relation of the DCTRS itself, which might be undecidable. This is in contrast to e.g. the definition of quasi-reductivity). Thus, in [73], [72], based on the idea of *unravelings* of [64, 63], a transformation from DCTRSs into TRSs is proposed such that termination of the transformed TRS implies *quasi-reductivity* of the given DCTRS which in turn implies its quasi-decreasingness.

We propose an alternative definition of *quasi-reductivity* using context-sensitivity ([55, 60]), that will be proved to be equivalent to operational termination of DCTRSs. Furthermore, we use a simple modification of Ohlebusch's transformation ([73]) that allows us to completely characterize the new property of context-sensitive quasi-reductivity of a DCTRS by means of termination of the context-sensitive (unconditional) TRS, that is obtained by the transformation, *on original terms* (i.e. terms over the signature of the DCTRS).

This complete characterization yields a method for disproving operational termination of DCTRSs by disproving termination of CSRSs on original terms. Moreover, we will show that the proposed transformation is sound and complete with respect to *collapse-extended* termination even if this notion is not restricted to original terms in the transformed system. As a corollary we obtain modularity of collapse-extended operational termination of DCTRSs.

Finally, we present an approach, which is based on the dependency pair framework of [33] (cf. also [9]), for proving termination of a CSRS on original terms, thus exploiting the given equivalence result. This approach has been implemented in the tool VMTL ([82])[1] and evaluated on a set of 24 examples. Several of these examples could be shown to be operationally terminating thanks to the new method, while other existing approaches fail.

3.2 Proving Operational Termination via Context-Sensitive Quasi-Reductivity

3.2.1 Context-Sensitive Quasi Reductivity

The first crucial step of our approach is the definition of *context-sensitive quasi-reductivity*, which will be proved to be equivalent to operational termination (cf., Corollary 4 below), and which is the key to the main results of this section.

Definition 1 (context-sensitive quasi-reductivity)**.** *A DCTRS $\mathcal{R}$ ($\mathcal{R} = (\mathcal{F}, R)$) is called* context-sensitively quasi-reductive *(*cs-quasi-reductive*) if there is an extension of the signature $\mathcal{F}'$ ($\mathcal{F}' \supseteq \mathcal{F}$), a replacement map μ (s.t. $\mu(f) = \{1, \dots, ar(f)\}$ for all $f \in \mathcal{F}$) and a μ-monotonic, well-founded partial order $\succ_\mu$ on $\mathcal{T}(\mathcal{F}', V)$ satisfying for every rule $l \to r \Leftarrow s_1 \to^* t_1, \dots, s_n \to^* t_n$, every substitution $\sigma\colon V \to \mathcal{T}(\mathcal{F}, V)$ and every $i \in \{0, \dots, n-1\}$:*

- *If $s_j\sigma \succeq_\mu t_j\sigma$ for every $1 \leq j \leq i$, then $l\sigma \succ_\mu^{st} s_{i+1}\sigma$.*
- *If $s_j\sigma \succeq_\mu t_j\sigma$ for every $1 \leq j \leq n$, then $l\sigma \succ_\mu r\sigma$.*

The ordering $\succ_\mu^{st}$ is defined as $(\succ_\mu \cup \rhd_\mu)^+$ where $t \rhd_\mu s$ if and only if s is a proper subterm of t at some position $p \in Pos^\mu(t)$. Moreover $\succeq_\mu = (\succ_\mu \cup =)$.

To be entirely precise, the notion of cs-quasi-reductivity should be parameterized by the set of function symbols that may not be restricted by the replacement map μ. However, as throughout the paper this set of function symbols is the set of functions of the signature of the DCTRS in question, we refrain from giving a reference to this parameter in the notion of *cs*-quasi-reductivity for the sake of simplicity.

Cs-quasi-reductivity generalizes quasi-reductivity in the sense that the extended signature may be equipped with a replacement map (which must leave the original signature untouched, though) and the monotonicity requirement of the ordering is relaxed accordingly. Furthermore, and this is crucial, in the ordering constraints for the conditional rules the substitutions replace variables only by terms over the

[1] `http://www.logic.at/vmtl/`

original signature, whereas in the original definition (of quasi-reductivity) terms over the extended signature are substituted.

The latter generalization appears to be quite natural, since the main implications of quasi-reductivity remain valid (cf. Proposition 2). Moreover, it is the key to the completeness results that we will prove (cf. Corollary 4).

Proposition 1. *If a DCTRS $\mathcal{R}$ is quasi-reductive, then it is cs-quasi-reductive.*

Proof. The result is obvious, since if a DCTRS is quasi-reductive with respect to a signature extension $\mathcal{F}'$ and an ordering $\succ$, then it is cs-quasi-quasi-reductive w.r.t. the same signature extension and the same ordering and the replacement map μ with $\mu(f) = \{1, \ldots, ar(f)\}$ for all $f \in \mathcal{F}'$. □

Proposition 2. *If a DCTRS $\mathcal{R}$ is cs-quasi-reductive, then it is quasi-decreasing.*

Proof. Let $\mathcal{R}$ be cs-quasi-reductive w.r.t. the ordering $\succ_\mu$. First, we show that $\rightarrow_\mathcal{R} \subseteq \succ_\mu$: Assume $s \rightarrow_\mathcal{R} t$ $(s, t \in \mathcal{T}(\mathcal{F}, V))$. We will use induction on the depth of the rewrite step in order to prove $s \succ_\mu t$. Assume the step $s \rightarrow_\mathcal{R} t$ has depth 1, i.e., an unconditional rule (or a rule with trivially satisfied conditions) is applied. In this case $s \succ_\mu t$ follows immediately from cs-quasi-reductivity of $\mathcal{R}$ and μ-monotonicity of $\succ_\mu$.

Next, assume the step $s \rightarrow_\mathcal{R} t$ has depth $d > 1$. Thus, a rule $l \rightarrow r \Leftarrow s_1 \rightarrow t_1, \ldots, s_n \rightarrow t_n$ is applied (i.e., $s|_p = l\sigma$). From the applicability of the conditional rule it follows that σ can be extended to σ' such that $s_i\sigma' \rightarrow^*_\mathcal{R} t_i\sigma'$ for all $1 \leq i \leq n$. Moreover, each reduction step in each of these reduction sequences has a depth smaller than d. Thus, the induction hypothesis and transitivity of $\succ_\mu$ yield $s_i\sigma' \succeq_\mu t_i\sigma'$ for all $1 \leq i \leq n$. Hence, by cs-quasi-reductivity we get $l\sigma' \succ_\mu r\sigma'$, and finally $s \succ_\mu t$ by μ-monotonicity of $\succ_\mu$.
Now we prove that $\mathcal{R}$ is quasi-decreasing w.r.t. the ordering $> := \succ_\mu^{st} |_{\mathcal{T}(\mathcal{F},V) \times \mathcal{T}(\mathcal{F},V)}$:

1. $\rightarrow_\mathcal{R} \subseteq >$: Follows immediately from $\rightarrow_\mathcal{R} \subseteq \succ_\mu \subseteq >$ if we restrict attention to terms of the original signature.

2. $> \; = \; >_{st}$: Assume there is a term s which is a proper subterm of a term $t \in \mathcal{T}(\mathcal{F}, V)$ $(t = C[s]_p)$, such that $t \not> s$. This implies $t \not\succ_\mu^{st} s$, which contradicts the definition of $\succ_\mu^{st}$ as p is a replacing position of t (because all positions in t are replacing).

3. For every rule $l \rightarrow r \Leftarrow s_1 \rightarrow t_1, \ldots, s_n \rightarrow t_n$, every substitution $\sigma \colon V \rightarrow \mathcal{T}(\mathcal{F}, V)$ and every $i \in \{0, \ldots, n-1\}$ we must show $s_j\sigma \rightarrow^* t_j\sigma$ for every $j \in \{1, \ldots, i\}$ implies $l\sigma > s_{i+1}\sigma$. We know that $s_j\sigma \rightarrow^* t_j\sigma \Rightarrow s_j\sigma \succeq_\mu t_j\sigma$. Because of cs-quasi-reductivity this implies $l\sigma \succ_\mu^{st} s_{j+1}\sigma$ and thus $l\sigma > s_{j+1}\sigma$, since $l\sigma, s_{j+1}\sigma \in \mathcal{T}(\mathcal{F}, V)$.

□

Corollary 1. *Let $\mathcal{R}$ be DCTRS. If $\mathcal{R}$ is cs-quasi-reductive, then it is* operationally terminating.

3.2.2 Verifying Context-Sensitive Quasi-Reductivity

In the following, we use a transformation from DCTRSs into CSRSs such that the μ-termination of the transformed CSRS implies cs-quasi-reductivity of the original DCTRS. The transformation is actually a variant of the one in [73], which in turn was inspired by [64, 63].

Definition 2 (unraveling of DCTRSs, [73])**.** *Let $\mathcal{R}$ be a DCTRS ($\mathcal{R} = (\mathcal{F}, R)$). For every rule $\alpha\colon l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n$ we use n new function symbols U_i^α ($i \in \{1, \ldots, n\}$). Then α is transformed into a set of unconditional rules in the following way:*

$$\begin{array}{rcl} l & \to & U_1^\alpha(s_1, \mathit{Var}(l)) \\ U_1^\alpha(t_1, \mathit{Var}(l)) & \to & U_2^\alpha(s_2, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1)) \\ & \vdots & \\ U_n^\alpha(t_n, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1), \ldots, \mathcal{E}\mathit{Var}(t_{n-1})) & \to & r \end{array}$$

Here, $\mathit{Var}(s)$ denotes an arbitrary but fixed sequence of the set of variables of the term s. Let $\mathcal{E}\mathit{Var}(t_i)$ be $\mathit{Var}(t_i) \setminus (\mathit{Var}(l) \cup \bigcup_{j=1}^{i-1} \mathit{Var}(t_j))$. By abuse of notation, by $\mathcal{E}\mathit{Var}(t_i)$ we denote an arbitrary but fixed sequence of the variables in the set $\mathcal{E}\mathit{Var}(t_i)$. Any unconditional rule of $\mathcal{R}$ is transformed into itself. The transformed system $U(\mathcal{R}) = (U(\mathcal{F}), U(R))$ is obtained by transforming each rule of $\mathcal{R}$ where $U(\mathcal{F})$ is $\mathcal{F}$ extended by all new function symbols. In case $\mathcal{R}$ has only one conditional rule α, we also write U_i instead of U_i^α.

Henceforth, we use the notion of U-symbols of a transformed signature, which are function symbols from $U(\mathcal{F}) \setminus \mathcal{F}$. Moreover, by U-terms or U-rooted terms we mean terms with a U-symbol as their root.

Next, we define the function tb, whose intended meaning is to undo non-finished meta-evaluations, i.e., evaluations of the form $s \to^*_{U(\mathcal{R})} U(v_1, \ldots, v_l)$. We call reductions of this shape *meta-evaluations*, because they are used for the evaluation of encoded conditions. This evaluation does not have an explicit counterpart in conditional rewrite sequences. The function tb and its properties will play a crucial role in understanding and proving the main results of this section.

Definition 3. *The mapping $\mathsf{tb}\colon \mathcal{T}(U(\mathcal{F}), V) \to \mathcal{T}(\mathcal{F}, V)$ (read "translate back") which is equivalent to Ohlebusch's mapping $\bigtriangledown$ ([73, Definition 7.2.53]) is defined by*

$$\mathsf{tb}(t) = \begin{cases} x & \textit{if } t = x \in V \\ f(\mathsf{tb}(t_1), \ldots, \mathsf{tb}(t_l)) & \textit{if } t = f(t_1, \ldots, t_l) \\ & \quad \textit{and } f \in \mathcal{F} \\ l\sigma & \textit{if } t = U_j^\alpha(v_1, v_2, \ldots, v_{k+1}, \ldots, v_{m_j}) \\ & \quad \textit{and } \alpha = l \to r \Leftarrow c \end{cases}$$

where $\mathit{Var}(l) = x_1, \ldots, x_k$ and σ is defined as $x_i\sigma = \mathsf{tb}(v_{i+1})$ for $1 \leq i \leq k$. Note that from Definition 2 it follows that $m_j \geq k + 1$.

Informally, the mapping tb translates back an evaluation of conditions to its start. Thus, $\mathsf{tb}(u) = u$ for every term $u \in \mathcal{T}(\mathcal{F}, V)$. Note that in general $s = \mathsf{tb}(t)$ does not imply $s \to^*_{U(\mathcal{R})} t$. The reason is that, for a term $t = U^\alpha_j(v_1, \ldots, v_l)$, the definition of $\mathsf{tb}(t)$ completely ignores the first argument t_1 of U^α_j.

Example 15. *Let $\mathcal{R}$ be a DCTRS consisting of one rule*

$$f(x) \;\to\; a \Leftarrow x \to b$$

$U(\mathcal{R})$ is given by the two rules

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

Consider the term $t = U(a,b)$. We have $\mathsf{tb}(t) = f(b)$ and clearly $f(b) \not\to^*_{U(\mathcal{R})} U(a,b)$.

Informally, the term $t = U^\alpha_j(v_1, \ldots, v_{m_j})$ represents an intermediate state of a reduction in $U(\mathcal{R})$ issuing from an original term, i.e., a term from $\mathcal{T}(\mathcal{F}, V)$, only if v_1 can be obtained (by reduction in $U(\mathcal{R})$) from the corresponding instance of the left-hand side of the corresponding condition of the applied conditional rule α.

The transformation of Definition 2 is suitable for verifying quasi-reductivity by proving termination of a TRS, as whenever the transformed system $U(\mathcal{R})$ is terminating, the original DCTRS $\mathcal{R}$ is *quasi-reductive* [73]. However, the converse implication does not hold.

Example 16. *([64]) Consider the DCTRS $\mathcal{R} = (\mathcal{F}, R)$ given by*

$$\begin{array}{lll} a \to c & c \to l & h(x,x) \to g(x,x,f(k)) \\ a \to d & d \to m & g(d,x,x) \to A \\ b \to c & k \to l & A \to h(f(a),f(b)) \\ b \to d & k \to m & \alpha\colon f(x) \to x \Leftarrow x \to^* e \\ c \to e & & \end{array}$$

The system $U(\mathcal{R}) = (U(\mathcal{F}), U(R))$ is given by $U(\mathcal{F}) = \mathcal{F} \cup \{U^\alpha_1\}$ and $U(R) = R$ except that rule α is replaced by the rules $f(x) \to U^\alpha_1(x,x)$ and $U^\alpha_1(e,x) \to x$. $\mathcal{R}$ is quasi-reductive (and thus operationally terminating) (cf. Example 20, below), nevertheless $U(\mathcal{R})$ is non-terminating ([73]).

Roughly speaking, the problem in Example 16 is that subterms at the second position of U^α_1 are reduced, while they are actually only supposed to "store" the variable bindings for future rewrite steps. These reductions can be prevented by using context-sensitivity. More precisely, we intend to forbid reductions of subterms which occur at or below a second, third, etc. argument position of an auxiliary U-symbol, according to the intuition that during the evaluation of conditions, the variable bindings should remain untouched. This leads to the following modification of the transformation, which has already been proposed independently by several authors (e.g., [20], [70], [18]) with slight differences.[2]

[2]See Section 3.4 (related work) for more details.

Definition 4. *(context-sensitive unraveling of a DCTRS) Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. The context-sensitive rewrite system $U_{cs}(\mathcal{R})$ uses the same signature and the same rules as $U(\mathcal{R})$. Additionally, a replacement map $\mu_{U_{cs}(\mathcal{R})}$ is used with $\mu_{U_{cs}(\mathcal{R})}(U) = \{1\}$ if $U \in U(\mathcal{F}) \setminus \mathcal{F}$ and $\mu_{U_{cs}(\mathcal{R})}(f) = \{1, \ldots, ar(f)\}$ if $f \in \mathcal{F}$.*

For notational simplicity we refer to $\mu_{U_{cs}(\mathcal{R})}$ just as μ if no confusion arises, e.g. in "μ-termination of $U_{cs}(\mathcal{R})$". Moreover, we omit an explicit reference to the replacement map $\mu_{U_{cs}(\mathcal{R})}$ if it is clear from the context, for instance in $\rightarrow_{U_{cs}(\mathcal{R})}$ reductions.

By using context-sensitivity in Definition 4 we get the following easy property of reduction sequences w.r.t. the obtained CSRSs.

Observation 1. *Let $\mathcal{R}$ be a DCTRS. For every reduction*

$$\begin{aligned} U_i(s_i\sigma', \vec{x}_i\sigma_i) &\overset{>\epsilon *}{\rightarrow}_{U_{cs}(\mathcal{R})} U_i(t_i\sigma'', \vec{x}_i\sigma_i) \\ &\overset{\epsilon}{\rightarrow}_{U_{cs}(\mathcal{R})} U_{i+1}(s_{i+1}\sigma''', \vec{x}_{i+1}\sigma_{i+1}) \end{aligned}$$

it holds that $x\sigma_i = x\sigma_{i+1}$ for all $x \in Dom(\sigma_i) \cap Dom(\sigma_{i+1})$.

In fact this is a crucial property of $U_{cs}(\mathcal{R})$, because given a DCTRS $\mathcal{R} = (\mathcal{F}, R)$ it guarantees that for each term $t \in \mathcal{T}(U(\mathcal{F}), V)$ we have $\mathsf{tb}(t) \rightarrow^*_{U_{cs}(\mathcal{R})} t$ provided that t is reachable by *any* term $s \in \mathcal{T}(\mathcal{F}, V)$ (see Corollary 2, below). This is in general not true, if context-sensitivity is dropped.

Example 17. *Let $\mathcal{R} = (\mathcal{F}, R)$ be the DCTRS of Example 15 extended by two unconditional rules*

$$\begin{aligned} f(x) &\rightarrow a \Leftarrow x \rightarrow b \\ a &\rightarrow b \\ a &\rightarrow c \end{aligned}$$

The transformed system $U(\mathcal{R})$ is

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

Consider the term $t = U(b, c)$. It is reachable in $U(\mathcal{R})$ from $f(a) \in \mathcal{T}(\mathcal{F}, V)$:

$$f(a) \rightarrow_{U(\mathcal{R})} U(a, a) \rightarrow_{U(\mathcal{R})} U(b, a) \rightarrow_{U(\mathcal{R})} U(b, c)$$

However, it is obviously not reachable from $\mathsf{tb}(t) = f(c)$ as b is not reachable from c. On the other hand, within $U_{cs}(\mathcal{R})$, $U(b, c)$ is not reachable by any *term from $\mathcal{T}(\mathcal{F}, V)$ because in $U_{cs}(\mathcal{R})$ reachability of a term t by any term $s \in \mathcal{T}(\mathcal{F}, V)$ (i.e. $s \rightarrow^*_{U_{cs}(\mathcal{R})} t$) implies reachability of t from $\mathsf{tb}(t)$ (cf. Corollary 2 below).*

The fact that in a CSRSs $U_{cs}(\mathcal{R})$ obtained by the context-sensitive transformation of a DCTRS $\mathcal{R} = (\mathcal{F}, R)$, each term t is reachable from $\mathsf{tb}(t)$ if t is part of a reduction sequence issuing from a term of $\mathcal{T}(\mathcal{F}, V)$, will be used extensively in the proofs of some of the main results of this section (e.g. Theorems 1 and 4).

Certain $U_{cs}(\mathcal{R})$-reduction steps inside a U-term t will have no effect on the result of the function tb, i.e., $t \to s$ and $\mathsf{tb}(t) = \mathsf{tb}(s)$. This motivates the definition of tb-preserving reduction steps in $U_{cs}(\mathcal{R})$. First, obviously reductions that occur strictly inside a U-term t do not alter the result of tb. The reason is that, because of context-sensitivity these reductions can only take place in the first argument of the root U-symbol and furthermore according to the definition of tb this first argument is irrelevant for the computation of tb.

Second, if a rule of the form $U_i^\alpha(s_1, \ldots, s_n) \to U_{i+1}^\alpha(t_1, \ldots, t_m)$ (whose right-hand side is a U-term) is applied to t, then tb applied to the resulting term also yields the same result as $\mathsf{tb}(t)$. The reason is that the variable bindings inside the U-term are preserved in such a step and all the variables that are present in l (where $\alpha = l \to r \Leftarrow c$) are already bound. For the same reason $\mathsf{tb}(t) = \mathsf{tb}(s)$ if t is not a U-term, s is a U-term and $t \to s$.

Definition 5 (tb-preserving reduction steps)**.** *Let $\mathcal{R}$ be a DCTRS ($\mathcal{R} = (\mathcal{F}, R)$) and $U_{cs}(\mathcal{R})$ its transformed CSRS. A step $s \xrightarrow{p}_{U_{cs}(\mathcal{R})} t$ is called* tb-preserving *if either p is strictly below some position q of s, where $root(s|_q)$ is a U-symbol, or $(t|_p)$ is a U-term.*

Proposition 3. *Let $\mathcal{R}$ be a DCTRS. If $s, t \in \mathcal{T}(U(\mathcal{F}), V)$ and $s \to_{U_{cs}(\mathcal{R})} t$ with a tb-preserving step, then $\mathsf{tb}(s) = \mathsf{tb}(t)$.*

Proof. If the reduction step from s to t, say at position p, occurs strictly inside a U-term, then it occurs strictly inside the first argument of some maximal U-rooted subterm u at position $q < p$ in s. According to Definition 3 we have $\mathsf{tb}(s|_q) = \mathsf{tb}(t|_q)$ and thus $\mathsf{tb}(t) = \mathsf{tb}(s)$.

Otherwise, $t|_p$ is a U-term. This means that either a rule of the shape $l \to U_1(c, \vec{x})$ or a rule of the shape $U_i(c_i, \vec{x}_i) \to U_{i+1}(c_{i+1}, \vec{x}_{i+1})$ was applied. So $s|_p = l\sigma$ and $t|_p = U_1(c, \vec{x})\sigma$, or $s|_p = U_i(c_i, \vec{x}_i)\sigma$ and $t|_p = U_{i+1}(c_{i+1}, \vec{x}_{i+1})\sigma$. Hence, according to Definition 3 we have $\mathsf{tb}(s|_p) = \mathsf{tb}(t|_p)$ and thus $\mathsf{tb}(s) = \mathsf{tb}(t)$. $\square$

Example 18. *Consider a CSRS $\mathcal{R}$*

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

with $\mu(U) = \mu(f) = \{1\}$. The following reductions are tb*-preserving:*

$$\begin{array}{rcl} \underline{f(a)} & \to_\mu & U(b, a), \text{ as } \mathsf{tb}(f(a)) = \mathsf{tb}(U(b, a)) = f(a) \\ U(\underline{b}, a) & \to_\mu & U(c, a), \text{ as } \mathsf{tb}(U(b, a)) = \mathsf{tb}(U(c, a)) = f(a) \end{array}$$

while the following one is not:

$$\underline{U(c,a)} \;\rightarrow_\mu\; a, \textit{ due to } \mathsf{tb}(U(c,a)) = f(a) \neq \mathsf{tb}(a) = a$$

Before investigating the effects of using context-sensitivity in the unraveling transformation of Definition 4 on the power of proving operational termination, let us consider the capability of $U_{cs}(\mathcal{R})$ to simulate reductions of a DCTRS $\mathcal{R}$. While *simulation completeness*, i.e., the property of $U_{cs}(\mathcal{R})$ of being able to mimic reductions of $\mathcal{R}$, is easy to obtain, *simulation soundness*, i.e., the property of $U_{cs}(\mathcal{R})$ to allow *only* those reductions (from original terms to original terms) that are also possible in $\mathcal{R}$, is non-trivial.

In [70] it was shown that simulation soundness is obtained for their version of the transformation if an additional restriction is imposed on reductions in $U_{cs}(\mathcal{R})$, which roughly states that only redexes without U-symbols (except at the root position) may be contracted. However, for our transformation this additional "membership condition" is not needed (see also Section 3.4 (related work) below for further details).

Theorem 1 (simulation completeness). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. For every $s, t \in \mathcal{T}(\mathcal{F}, V)$ we have: if $s \rightarrow_{\mathcal{R}} t$, then $s \rightarrow^+_{U_{cs}(\mathcal{R})} t$.*

Proof. We use induction on the depth of the step $s \rightarrow_{\mathcal{R}} t$. If $s \rightarrow_{\mathcal{R}} t$ with a rule $l \rightarrow r$ (i.e., an unconditional rule), then $l \rightarrow r \in U_{cs}(\mathcal{R})$ and thus $s \rightarrow_{U_{cs}(\mathcal{R})} t$. Assume $s \rightarrow_{\mathcal{R}} t$ with a rule $\alpha\colon l \rightarrow r \Leftarrow s_1 \rightarrow^* t_1, \ldots, s_n \rightarrow^* t_n$. Then $s = C[l\sigma]_p$ and $t = C[r\sigma]_p$. All rewrite sequences $s_i\sigma \rightarrow^*_{\mathcal{R}} t_i\sigma$ have lower depths than $l\sigma \rightarrow_{\mathcal{R}} r\sigma$, thus we can apply the induction hypothesis to obtain the following rewrite sequence in the transformed system:

$$\begin{array}{rll}
C[l\sigma]_p & \rightarrow_{U_{cs}(\mathcal{R})} & C[U_1^\alpha(s_1\sigma,\, \mathit{Var}(l)\sigma)]_p \\
& \rightarrow^*_{U_{cs}(\mathcal{R})} & C[U_1^\alpha(t_1\sigma,\, \mathit{Var}(l)\sigma)]_p \\
& \rightarrow_{U_{cs}(\mathcal{R})} & C[U_2^\alpha(s_2\sigma,\, \mathit{Var}(l)\sigma, \mathcal{E}\mathit{Var}(t_1)\sigma)]_p \\
& \rightarrow^*_{U_{cs}(\mathcal{R})} & \cdots \\
& \rightarrow^*_{U_{cs}(\mathcal{R})} & C[U_n^\alpha(t_n\sigma,\, \mathit{Var}(l)\sigma, \mathcal{E}\mathit{Var}(t_1)\sigma, \ldots, \mathcal{E}\mathit{Var}(t_{n-1})\sigma)]_p \\
& \rightarrow_{U_{cs}(\mathcal{R})} & C[r\sigma]_p = t
\end{array}$$

□

Theorem 2 (simulation soundness). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. For every $s, t \in \mathcal{T}(U(\mathcal{F}), V)$ we have: If $s \rightarrow^*_{U_{cs}(\mathcal{R})} t$ and s is reachable from an original term (i.e., $s' \rightarrow^*_{U_{cs}(\mathcal{R})} s$ for some $s' \in \mathcal{T}(\mathcal{F}, V)$), then $\mathsf{tb}(s) \rightarrow^*_{\mathcal{R}} \mathsf{tb}(t)$. Moreover, if $s, t \in \mathcal{T}(\mathcal{F}, V)$, then $s \rightarrow^+_{U_{cs}(\mathcal{R})} t$ implies $s \rightarrow^+_{\mathcal{R}} t$.*

Before proving Theorem 2 we need two auxiliary lemmas. The first one (Lemma

1 below) states that whenever we have a $U_{cs}(\mathcal{R})$-reduction sequence D of the shape

$$\begin{array}{lll} s_1 & \to^*_{U_{cs}(\mathcal{R})} & s_2[l\sigma]_{p_1} \\ & \to_{U_{cs}(\mathcal{R})} & s_2[U_1^\alpha(s_1, \vec{x}_1)\sigma]_{p_1} \\ & \to^*_{U_{cs}(\mathcal{R})} & s_3[U_1^\alpha(t_1, \vec{x}_1)\sigma']_{p_2} \\ & \to_{U_{cs}(\mathcal{R})} & s_3[U_2^\alpha(s_2, \vec{x}_2)\sigma']_{p_2} \\ & \to^*_{U_{cs}(\mathcal{R})} & \cdots \\ & \to^*_{U_{cs}(\mathcal{R})} & s_{n+1}[U_n^\alpha(t_n, \vec{x}_n)\sigma^n]_{p_n} \\ & \to_{U_{cs}(\mathcal{R})} & s_{n+1}[r\sigma^n]_{p_n} \\ & \to^*_{U_{cs}(\mathcal{R})} & s_{n+2}, \end{array}$$

where s_1 is an original term which means that D contains the complete simulation of the application of a conditional rule $\alpha\colon l \to r \Leftarrow s_1 \to^* t_1, \dots s_n \to^* t_n$, the reductions satisfying its conditions $s_i\sigma^n \to^*_{U_{cs}(\mathcal{R})} t_i\sigma^n$ occur as subreductions of D for all $i \in \{1, \dots, n\}$.

The second auxiliary result (Lemma 2 below) will be crucial for the overall inductive proof structure of Theorem 2 to work.

For the former we first introduce some terminology for tracing subterms (especially U-subterms) in reduction sequences, in a forward and backward manner. In the above reduction sequence D the positions p_i mark *descendants* of the subterms $s_{i-1}|_{p_{i-1}}$ of s_{i-1}. More formally, the set of *one-step descendants* of a subterm position p of t w.r.t. a (one-step) reduction $t = C[s]_p \xrightarrow{q} t'$ is the set of subterm positions in t' given by

- $\{p\}$, if $q \geq p$ or $q \mid\mid p$,
- $\{q.o'.p' \mid t|_q = l\sigma, l|_o \in \mathit{Var}, q.o.p' = p, l|_o = r|_{o'}\}$, if $q < p$ and (a superterm of) s is bound to a variable in the matching of $t|_q$ with the left-hand side of the applied rule, and
- $\emptyset$, otherwise.

Slightly abusing terminology, when $t = C[s]_p \xrightarrow{q} t'$ with set $\{p_1, \dots, p_k\}$ of one-step descendants in t', we also say that $t|_p$ has descendants $t'|_{p_i}$ in t'. The *descendant relation* (w.r.t given derivations) is obtained as the (reflexive-)transitive closure of the one-step descendant relation. Note that the set of one-step descendants of a U-subterm (w.r.t. a one-step derivation) is non-empty unless the subterm is erased by an erasing rule (i.e., a rule $l \to r$ such that $x \in \mathit{Var}(l) \setminus \mathit{Var}(r)$), because U-symbols occur only at but not below the root position in left-hand sides of rules of systems obtained by the transformation of Definition 4. The notions of *one-step* (and many-step) *ancestors* of a subterm position (w.r.t. a given reduction sequence) are defined analogously (in a backward manner).

Note that with a similar argument as for the existence of descendants of U-subterms we get that every U-subterm has at least one one-step ancestor w.r.t. every (one-step) reduction sequence.

Now we can express the notion of a complete simulated rule application more formally. By a complete simulated rule application we mean that all rules obtained by transforming one conditional rule are eventually applied to a certain subterm and its descendants during the reduction sequence in the right order. Yet, these (unconditional) rule applications need not be consecutive.

Note also that it makes sense to talk about descendants of U-subterms, because they can only be copied, eliminated or duplicated but not otherwise modified by more outer reductions. This is due to the special shape of the rules in systems obtained by the transformation of Definition 4. More precisely, it is due to the fact that U-symbols occur only at, but not below the root of left-hand sides of rules.

Lemma 1. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS and let $\alpha\colon l \to r \Leftarrow c_1^l \to^* c_1^r, \dots c_n^l \to^* c_n^r$ be a rule from R. Moreover, assume that $D\colon s \to^*_{U_{cs}(\mathcal{R})} t$ is a non-empty $U_{cs}(\mathcal{R})$-reduction such that $s \in \mathcal{T}(\mathcal{F}, V)$ and the last step is due to an application of the rule $U_n^\alpha(c_n^r, \vec{x}_n) \to r \in U_{cs}(\mathcal{R})$ ($r \in \mathcal{T}(\mathcal{F}, V)$) with a substitution σ^n. Then, the reductions $C_i\colon c_i^l\sigma^n \to^*_{U_{cs}(\mathcal{R})} c_i^r\sigma^n$ occur as subreductions of D[3] for every $i \in \{1, \dots, n\}$.*

Proof. Let the last step of D be $t' \xrightarrow{p}_{U_{cs}(\mathcal{R})} t$. Hence, $t'|_p$ must be a U-term.

We identify the first term s' in D such that

1. s' contains at least one ancestor of $t'|_p$,

2. all ancestors of $t'|_p$ in s' are U-terms, and

3. conditions (1) and (2) also hold for all terms occurring later (but before t') in D.

Note that t' itself has the demanded properties. Thus the existence of s' is guaranteed. We now claim:

$$\text{Some ancestor } s'|_q \text{ of } t'|_p \text{ has the form } U_1^\alpha(c_1^l, \vec{x}_1)\sigma^1. \tag{3.1}$$

In order to show (3.1) assume s' did not contain a subterm of this shape. Then, consider s^0 which is the term occurring immediately before s' in D (this subterm exists as s' contains U-terms but s does not, so $s \neq s'$). The term s^0 contains ancestors of $t'|_p$, because s' contains ancestors of $t'|_p$ which are U-terms. This in turn implies the existence of one-step ancestors of $t'|_p$ in s^0.

Assume some ancestor of $t'|_p$ in s^0 is not a U-term. As this term has a one-step descendant in s' being in turn an ancestor of $t'|_p$ and thus a U-term, this very U-term in s' must be of the shape $U_1^\alpha(c_1^l, \vec{x}_1)\sigma^1$ as it must have been introduced by an

[3]not necessarily consecutively, and embedded in some surrounding context, i.e. they can be obtained by "extraction" from D

application of the rule $l \to U_1^\alpha(c_1^l, \vec{x}_1)$. This contradicts our assumption that (3.1) does not hold for s'. Thus all ancestors of $t'|_p$ in s^0 must be U-terms.

This in turn contradicts the minimality of s', i.e. being the first term in D containing only U-term ancestors of $t'|_p$. Hence, we derived a contradiction from $\neg$(3.1). This concludes the proof of Claim 3.1.

Let $s'|_q = U_1^\alpha(c_1^l, \vec{x}_1)\sigma^1$. By our choice of s' and the fact that $s'|_q$ is an ancestor of $t'|_p$, every term between s' and t' in D contains a descendant of $s'|_q$ which is also an ancestor of $t'|_p$ and a U-term.

Some descendant (of $s'|_q$) must be of the shape $U_1^\alpha(c_1^r, \vec{x}_1)\sigma'^1$, because otherwise $t'|_p$ could not be reached (cf. Definition 4). We inspect D between s' and s'' where s'' contains such a descendant of $s'|_q$ say at position q'. Then, $s'|_q$ and its descendants which are also ancestors of $s''|_{q'}$ are only (syntactically) modified by rule applications below their roots. The reason is that a term rooted by some U-symbol U_i cannot be reduced to another term having the same root symbol with reduction steps containing at least one root step, unless the reduction sequence contains a non-U-term (cf. Definition 4).

Hence, we can extract the reduction $c_1^l\sigma^1 \to^*_{U_{cs}(\mathcal{R})} c_1^r\sigma'^1$ from D.

The same argumentation applies also to all other conditions as $U_i^\alpha(c_i^l, \vec{x}_i)\sigma^i$ must occur (by our choice of s' and q), as descendants of $s'|_q$ and ancestors of $t'|_p$ in D (in particular, in such a way that σ^i does not contradict σ'^{i-1}). Moreover, by Observation 1 the used substitutions are not contradictory and their domains are subdomains of the one of σ^n, which is due to the fact that the set of variables stored by a U-symbol U_i^α is a subset of the ones stored by U_j^α provided that $i \leq j$ (cf. Definition 4). □

The second lemma states that the existence of a parallel reduction sequence $s \mathrel{+\!\!\!|\!\!\!\!\!\rightarrow}^*_{U_{cs}(\mathcal{R})} t$, where s is an original term, implies that for all positions p of t there is also a parallel reduction $s' \mathrel{+\!\!\!|\!\!\!\!\!\rightarrow}^*_{U_{cs}(\mathcal{R})} t|_p$ for some original term s' such that its length is less than or equal to the length of the former parallel reduction sequence.

In order to formalize this proposition we introduce the notion of the *minimal parallel $\mathcal{F}$-distance* of a term (over $\mathcal{T}(U(\mathcal{F}), V)$) (from any original term).

Definition 6 (minimal parallel $\mathcal{F}$-distance)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS and $t \in \mathcal{T}(U(\mathcal{F}), V)$. The* minimal parallel $\mathcal{F}$-distance *of t (w.r.t. a DCTRS $\mathcal{R}$) is given by*

$$mpd_{\mathcal{F}}(t) = inf\{n \mid \exists s \in \mathcal{T}(\mathcal{F}, V). s \mathrel{+\!\!\!|\!\!\!\!\!\rightarrow}^n_{U_{cs}(\mathcal{R})} t\}$$

where $\mathrel{-\!\!\!+\!\!\!|\!\!\!\!\!\rightarrow}^n_{U_{cs}(\mathcal{R})}$ means that n parallel reductions are performed and inf is the infimum.

Note that $inf\ \emptyset = +\infty$, so the minimal parallel $\mathcal{F}$-distance of any term t that is not reachable from an original term is $+\infty$. Note on the other hand that if t is reachable from an original term, then the infimum in Definition 6 is actually a minimum, as lengths of reductions are natural numbers and hence we can find a concrete (parallel) reduction from an original term to t with length $mpd_{\mathcal{F}}(t)$.

Lemma 2. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS and $t \in \mathcal{T}(U(\mathcal{F}), V)$ with $mpd_{\mathcal{F}}(t) < \infty$. Then for every subterm $t|_p$ of t we have $mpd_{\mathcal{F}}(t|_p) \leq mpd_{\mathcal{F}}(t)$. Moreover, if $t|_p$ occurs strictly inside a U-term in t, then $mpd_{\mathcal{F}}(t|_p) < mpd_{\mathcal{F}}(t)$.*

Proof. Let $D\colon u \mathrel{-\!\!\!\|\!\!\!\rightarrow}^*_{U_{cs}(\mathcal{R})} t$ where $u \in \mathcal{T}(\mathcal{F}, V)$ be a reduction sequence of length $mpd_{\mathcal{F}}(t)$. We prove the result by induction on $mpd_{\mathcal{F}}(t)$. If $mpd_{\mathcal{F}}(t) = 0$, the result holds vacuously as t is an original term and thus every subterm of t is an original term as well.

Assume $mpd_{\mathcal{F}}(t) = m$, then we can write D as

$$u \mathrel{-\!\!\!\|\!\!\!\rightarrow}^{m-1}_{U_{cs}(\mathcal{R})} t' \mathrel{-\!\!\!\|\!\!\!\rightarrow}_{U_{cs}(\mathcal{R})} t.$$

We consider the maximal U-rooted subterms $u_1, \ldots, u_n$ of $t|_p$ s.t.

$$t|_p = C[u_1, \ldots u_n]_{p_1,\ldots,p_n}.$$

Each subterm u_i has at least one one-step ancestor u'_i in t' and the induction hypothesis yields that $mpd_{\mathcal{F}}(u'_i) \leq mpd_{\mathcal{F}}(t') \leq m - 1$. Hence, as we are using parallel reduction we obtain

$$mpd_{\mathcal{F}}(C[u'_1, \ldots u'_n]_{p_1,\ldots,p_n}) \leq m - 1. \tag{3.2}$$

Moreover, we have $u'_i \mathrel{-\!\!\!\|\!\!\!\rightarrow}^*_{U_{cs}(\mathcal{R})} u_i$ with zero or one reduction steps and as all u_i are parallel in $t|_p$ we have $C[u'_1, \ldots u'_n]_{p_1,\ldots,p_n} \mathrel{-\!\!\!\|\!\!\!\rightarrow}^*_{U_{cs}(\mathcal{R})} C[u_1, \ldots u_n]_{p_1,\ldots,p_n} = t|_p$ with zero or one steps. Thus, $mpd_{\mathcal{F}}(t|_p) \leq m$.

Now assume that $t|_p$ occurs strictly inside a U-term in t. We distinguish two cases.

- If $u'_i \neq u_i$ for some $i \in \{1, \ldots, n\}$ (i.e. if some reduction step from t' to t occurred inside an ancestor of some u_i), then by the definition of the descendant relation and the shape of the rules in $U_{cs}(\mathcal{R})$, i.e. the fact that U-symbols occur only at but not below the root of left- and right-hand sides of all rules, we get that if u_i occurs at position $q \geq p$ in t, then u'_i occurs at position q in t' and there must have been a reduction in $t' \mathrel{-\!\!\!\|\!\!\!\rightarrow}_{U_{cs}(\mathcal{R})} t$ at or below q. Hence, in the same parallel step there was no reduction above p and thus all u'_i occur inside a U-term in t' as they occur inside $t'|_p$.

  Hence, the induction hypothesis yields $mpd_{\mathcal{F}}(u'_i) \leq m - 2$ and

  thus we get $mpd_{\mathcal{F}}(t|_p) \leq m - 1$.

- Otherwise, if $u'_i = u_i$ for all $i \in \{1, \ldots, n\}$, (3.2) and $C[u'_1, \ldots u'_n]_{p_1,\ldots,p_n} = t|_p$ yield $mpd_{\mathcal{F}}(t|_p) \leq m - 1$.

$\square$

Indeed, Lemma 2 does not hold if one considers ordinary $U_{cs}(\mathcal{R})$-reduction instead of parallel reduction.

Example 19. *Consider the following one-rule DCTRS $\mathcal{R}$*

$$f(x) \to b \Leftarrow g(x,x) \to^* a$$

$U_{cs}(\mathcal{R})$ is given by

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

Now consider the following $U_{cs}(\mathcal{R})$ reduction sequence of length 2

$$\begin{array}{rcl} f(f(x)) & \to & f(U(g(x,x),x)) \\ & \to & U(g(U(g(x,x),x),U(g(x,x),x)),U(g(x,x),x)) = t \end{array}$$

However, it is easy to see that at least 2 reduction steps are necessary to derive

$$t|_1 = g(U(g(x,x),x),U(g(x,x),x))$$

from an original term although it occurs as subterm strictly below a U-symbol in t.

Now we are ready to prove Theorem 2.

Proof of Theorem 2. For the first part of the theorem, we prove the equivalent result that $s \mathrel{+\!\!\!\!\mapsto}^*_{U_{cs}(\mathcal{R})} t$ implies $\mathsf{tb}(s) \mathrel{+\!\!\!\!\mapsto}^*_{\mathcal{R}} \mathsf{tb}(t)$ provided that $s,t \in \mathcal{T}(U(\mathcal{F}),R)$ and s is reachable from an original term.

In order to prove this by induction, we associate to each reduction sequence $S\colon s \mathrel{+\!\!\!\!\mapsto}^*_{U_{cs}(\mathcal{R})} t$ with $s,t \in \mathcal{T}(U(\mathcal{F}),V)$ a non-negative integer (its order) k where $k = mpd_{\mathcal{F}}(s)+l$ and l is the length (i.e. the number of parallel reduction steps) of S. We use induction over k (note that $mpd_{\mathcal{F}}(s)$ and l are both non-negative for every reduction sequence S).

For the base case (i.e., $k=0$) the theorem holds trivially, since $s=t$. For the inductive step, consider a reduction sequence $S\colon s \mathrel{+\!\!\!\!\mapsto}^*_{U_{cs}(\mathcal{R})} s' \mathrel{+\!\!\!\!\mapsto}_{U_{cs}(\mathcal{R})} t$. The induction hypothesis yields $\mathsf{tb}(s) \to^*_{\mathcal{R}} \mathsf{tb}(s')$. Thus, for $\mathsf{tb}(s) \to^*_{\mathcal{R}} \mathsf{tb}(t)$ it suffices to show that

$$\mathsf{tb}(s') \to^*_{\mathcal{R}} \mathsf{tb}(t) \tag{3.3}$$

holds.

We prove this by (nested) induction over the number of single (non-parallel) reduction steps in $s' \mathrel{+\!\!\!\!\mapsto}_{U_{cs}(\mathcal{R})} t$. If this number is zero, then $s' = t$ and thus $\mathsf{tb}(s') = \mathsf{tb}(t)$.

Otherwise, we split $s' \mathrel{+\!\!\!\!\mapsto}_{U_{cs}(\mathcal{R})} t$ into $s' \mathrel{+\!\!\!\!\mapsto}_{U_{cs}(\mathcal{R})} t' \to_{U_{cs}(\mathcal{R})} t$ and the induction hypothesis yields $\mathsf{tb}(s') \to^*_{\mathcal{R}} \mathsf{tb}(t')$.

We distinguish 3 cases depending on the reduction from t' to t.

1. Assume the step is tb-preserving. Then we have $\mathsf{tb}(t') = \mathsf{tb}(t)$ and hence $\mathsf{tb}(s') \to^*_{\mathcal{R}} \mathsf{tb}(t)$, i.e., (3.3).

2. If the step is non-tb-preserving and using a rule $l \to r$ which already occurred in the DCTRS (i.e. as unconditional rule) say at position p, then $t'|_p = l\sigma$. As the reduction is non-tb-preserving, there is no U-symbol in t' above p (cf. Definition 5). Moreover, there are no U-symbols in l (as it already occurred in $\mathcal{R}$), hence $\mathsf{tb}(t')|_{p.q} = \mathsf{tb}(t'|_{p.q})$ for all variable positions q of l, i.e. $\mathsf{tb}(t')|_p = l\sigma'$ and $x\sigma' = \mathsf{tb}(x\sigma)$ for all $x \in \mathit{Dom}(\sigma)$. Thus, $\mathsf{tb}(t') = \mathsf{tb}(t')[l\sigma']_p \to_{\mathcal{R}} \mathsf{tb}(t')[r\sigma']_p = \mathsf{tb}(t)$, and finally (3.3).

3. Assume the step (at position p) is non-tb-preserving and using a rule
$$U(u, x_1, \ldots, x_o) \to r$$
where $\mathit{root}(r) \in \mathcal{F}$ (say $t'|_p = U(u, x_1, \ldots, x_o)\sigma)$. This rule stems from a conditional rule $\alpha\colon l \to r \Leftarrow c^l_1 \to^* c^r_1, \ldots c^l_m \to^* c^r_m \in \mathcal{R}$.

   In order to perform the corresponding reduction in the conditional system $\mathcal{R}$, we need to make sure that $\mathsf{tb}(c^l_i\sigma) \to^*_{\mathcal{R}} \mathsf{tb}(c^r_i\sigma)$ holds for every $i \in \{1, \ldots, m\}$.

   We consider the following reduction sequence S' in $U_{cs}(\mathcal{R})$
$$S'\colon u \mathrel{+\!\!\!\!\!\!\;|\!\!\!\to}^*_{U_{cs}(\mathcal{R})} s \mathrel{+\!\!\!\!\!\!\;|\!\!\!\to}^*_{U_{cs}(\mathcal{R})} s'$$
where u is some *original* term such that the length of the reduction from u to s is exactly $\mathit{mpd}_{\mathcal{F}}(s)$. Note that $s'|_p = U(u, x_1, \ldots, x_o)\sigma)$, because all reduction steps from s' to t' were parallel to p.

   The existence of S' ensures that for each condition $c^l_i \to^* c^r_i$ the reduction $c^l_i\sigma \mathrel{+\!\!\!\!\!\!\;|\!\!\!\to}^*_{U_{cs}(\mathcal{R})} c^r_i\sigma$ occurred as subreduction of S', by Lemma 1.

   Consider a term $c^l_i\sigma$ occurring as a subterm of some term v in S'. We partition the reduction sequence S' in reduction steps that happen before v (which we call the head of S') and in reduction steps happening after v (which we call the tail of S').

   The reduction from $c^l_i\sigma$ to $c^r_i\sigma$ is part of the tail of S' and thus its (parallel) length is not longer than this tail. Moreover, Lemma 2 yields that $\mathit{mpd}_{\mathcal{F}}(c^l_i\sigma)$ is shorter than the head of S', because $c^l_i\sigma$ occurs inside a U-term. Hence, the order of the reduction sequence $c^l_i\sigma \mathrel{+\!\!\!\!\!\!\;|\!\!\!\to}^*_{U_{cs}(\mathcal{R})} c^r_i\sigma$ is smaller than (or equal to) the length of the reduction sequence S' which is exactly the order of the reduction from s to s' and thus smaller than the order of our initial reduction sequence S. Hence, the induction hypothesis (of the outer induction) applies yielding $\mathsf{tb}(c^l_i\sigma) \to^*_{\mathcal{R}} \mathsf{tb}(c^r_i\sigma)$ for all $i \in \{1, \ldots, m\}$.

   Now consider $t' = t'[U(u, x_1, \ldots, x_o)\sigma]_p$. Let $\tau = \mathsf{tb}(\sigma)$, i.e. $x\tau = \mathsf{tb}(x\sigma)$ for all $x \in \mathit{Dom}(\sigma)$. Then we have $\mathsf{tb}(t') = \mathsf{tb}(t')[l\tau]_p$. And since $c^l_i\tau \to^*_{\mathcal{R}} c^r_i\tau$ for all $i \in \{1, \ldots, m\}$, we finally obtain $\mathsf{tb}(t')[l\tau]_p \to_{\mathcal{R}} \mathsf{tb}(t')[r\tau]_p = \mathsf{tb}(t)$.

This concludes the inner induction and also the outer step case.

Note that, in the inner induction above, if not all steps in a reduction sequence S are tb-preserving, i.e. whenever items (2) or (3) apply, then the corresponding sequence in the conditional system is non-empty. Hence, whenever $s \rightarrow^{+}_{U_{cs}(\mathcal{R})} t$ and $s, t \in \mathcal{T}(\mathcal{F}, V)$, then $\mathsf{tb}(s) \rightarrow^{+}_{\mathcal{R}} \mathsf{tb}(t)$ is non-empty, too. □

Next we show that for any term t that is reachable from an original one, say s, the corresponding reduction can be factored through $\mathsf{tb}(t)$ such that the first part only uses $\mathcal{R}$-steps and the latter one only tb-preserving $U_{cs}(\mathcal{R})$-steps.

Lemma 3. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. If a term $t \in \mathcal{T}(U(\mathcal{F}), V)$ is reachable from an original term (i.e., if $mpd_{\mathcal{F}}(t) < \infty$), then we have $\mathsf{tb}(t) \rightarrow^{*}_{U_{cs}(\mathcal{R})} t$ with tb-preserving steps.*

Proof. We prove the result by induction on $mpd_{\mathcal{F}}(t)$. If $mpd_{\mathcal{F}}(t) = 0$, then t is an original term and the result is immediate.

Otherwise, let $mpd_{\mathcal{F}}(t) = n > 0$. Then, there is a parallel $U_{cs}(\mathcal{R})$-reduction sequence $D\colon u \mathrel{+\!\!\!\!\!\!\rightarrow}^{n-1}_{U_{cs}(\mathcal{R})} t' \mathrel{+\!\!\!\!\!\!\rightarrow}_{U_{cs}(\mathcal{R})} t$ of length n with $u \in \mathcal{T}(\mathcal{F}, V)$. Let $u_1, \ldots, u_m$ be the maximal U-rooted subterms of t s.t.

$$t = C[u_1, \ldots, u_m]_{p_1, \ldots, p_m}$$

Each u_i has one or several one-step ancestors u_i^j (in t') for $j \in \{1, \ldots, k_i\}$, where k_i is the number of one-step ancestors of u_i in D. For all $i \in \{1, \ldots, m\}$ and all $j \in \{1, \ldots, k_i\}$ $mpd_{\mathcal{F}}(u_i^j) < n$ by Lemma 2, hence the induction hypothesis yields $\mathsf{tb}(u_i^j) \rightarrow^{*}_{U_{cs}(\mathcal{R})} u_i^j$ with tb-preserving steps.

Moreover, we get $u_i^j \rightarrow^{0/1}_{U_{cs}(\mathcal{R})} u_i$ for all $i \in \{1, \ldots, m\}$ and all $j \in \{1, \ldots, k_i\}$, and these steps are tb-preserving, because the u_i's are U-terms. Hence, we obtain $\mathsf{tb}(u_i) = \mathsf{tb}(u_i^j) \rightarrow^{*}_{U_{cs}(\mathcal{R})} u_i^j \rightarrow^{0/1}_{U_{cs}(\mathcal{R})} u_i$ with tb-preserving steps and as the u_i's are the maximal U-rooted terms in t, we finally get

$$\mathsf{tb}(t) = C[\mathsf{tb}(u_1), \ldots, \mathsf{tb}(u_m)]_{p_1, \ldots, p_m} \rightarrow^{*}_{U_{cs}(\mathcal{R})} C[u_1, \ldots, u_m]_{p_1, \ldots, p_m} = t$$

with only tb-preserving steps. □

Corollary 2. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. Whenever $s \rightarrow^{*}_{U_{cs}(\mathcal{R})} t$ and $t, s \in \mathcal{T}(U(\mathcal{F}), V)$ where s is reachable from an original term, $\mathsf{tb}(s) \rightarrow^{*}_{U_{cs}(\mathcal{R})} \mathsf{tb}(t) \rightarrow^{*}_{U_{cs}(\mathcal{R})} t$, such that $\mathsf{tb}(t) \rightarrow^{*}_{U_{cs}(\mathcal{R})} t$ consists only of tb-preserving steps.*

Proof. Immediate consequence of Theorems 2, 1 and Lemma 3. □

Regarding termination, the transformation of Definition 4 is sound for cs-quasi-reductivity in the sense that μ-termination of $U_{cs}(\mathcal{R})$ implies context-sensitive quasi-reductivity and thus operational termination of $\mathcal{R}$.

Theorem 3 (sufficiency for cs-quasi-reductivity)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. If $U_{cs}(\mathcal{R})$ is μ-terminating, then $\mathcal{R}$ is cs-quasi-reductive.*

Proof. As $U_{cs}(\mathcal{R})$ is $\mu_{U_{cs}(\mathcal{R})}$-terminating, $\succ_\mu = \rightarrow^+_{U_{cs}(\mathcal{R})}$ is a μ-reduction ordering on $\mathcal{T}(U(\mathcal{F}), V)$ (where $U(\mathcal{F}) \supseteq \mathcal{F}$). Assume $s_j\sigma \succeq_\mu t_j\sigma$ for every $1 \leq j \leq i < n$ for a rule $\alpha\colon l \rightarrow r \Leftarrow s_1 \rightarrow^* t_1, \dots, s_n \rightarrow^* t_n$ ($\sigma\colon V \rightarrow \mathcal{T}(\mathcal{F}, V)$). Then we have the following reduction sequence in $U_{cs}(\mathcal{R})$:

$$\begin{array}{lll} l\sigma & \rightarrow_{U_{cs}(\mathcal{R})} & U_1^\alpha(s_1, \mathit{Var}(l))\sigma \\ & \rightarrow^*_{U_{cs}(\mathcal{R})} & U_1^\alpha(t_1, \mathit{Var}(l))\sigma \\ & \rightarrow_{U_{cs}(\mathcal{R})} & U_2^\alpha(s_2, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1))\sigma \\ & \rightarrow^*_{U_{cs}(\mathcal{R})} & U_2^\alpha(t_2, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1))\sigma \\ \dots & & \\ & \rightarrow_{U_{cs}(\mathcal{R})} & U_i^\alpha(s_i, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1), \dots, \mathcal{E}\mathit{Var}(t_{i-1}))\sigma \\ & \rightarrow^*_{U_{cs}(\mathcal{R})} & U_i^\alpha(t_i, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1), \dots, \mathcal{E}\mathit{Var}(t_{i-1}))\sigma \\ & \rightarrow_{U_{cs}(\mathcal{R})} & U_{i+1}^\alpha(s_{i+1}, \mathit{Var}(l), \mathcal{E}\mathit{Var}(t_1), \dots, \mathcal{E}\mathit{Var}(t_i))\sigma \end{array}$$

Thus $l\sigma \succ^{st}_\mu s_{i+1}\sigma$. If $s_j\sigma \succeq_\mu t_j\sigma$ for all $1 \leq j \leq n$, then it is easy to see that there is a reduction sequence $l\sigma \rightarrow^+_{U_{cs}(\mathcal{R})} r\sigma$, thus $l\sigma \succ_\mu r\sigma$. □

The following corollary (of Theorem 3 and Corollary 1) has already been proved in [18].

Corollary 3. *([18]) Let $\mathcal{R}$ be a DCTRS. If $U_{cs}(\mathcal{R})$ is μ-terminating, then $\mathcal{R}$ is operationally terminating.*

Obviously, as $U(\mathcal{R})$ and $U_{cs}(\mathcal{R})$ differ only in that $U_{cs}(\mathcal{R})$ uses an additional replacement map, the context-sensitive transformation is more powerful when it comes to verifying operational termination.

Proposition 4. *([18]) Let $\mathcal{R}$ be a DCTRS. If $U(\mathcal{R})$ is terminating, then $U_{cs}(\mathcal{R})$ is μ-terminating.*

Proof. The result is immediate, since we have $\rightarrow_{U(\mathcal{R})} \supseteq \rightarrow_{U_{cs}(\mathcal{R})}$. □

Example 20. *Consider the DCTRS $\mathcal{R}$ of Example 16. The transformed system $U_{cs}(\mathcal{R})$ (which is identical to the non-terminating TRS $U(\mathcal{R})$, except for the fact that an additional replacement map is used) is μ-terminating. This can for instance be proved by minimal counterexample and case analysis. However, we will see that in order to verify operational termination of $\mathcal{R}$, it is sufficient to prove a weaker form of termination, which can be handled automatically (see Theorem 5 and Example 25 below).*

Unfortunately, and interestingly, cs-quasi-reductivity of a DCTRS $\mathcal{R}$ does not imply μ-termination of $U_{cs}(\mathcal{R})$.

Example 21. *([73, Ex. 7.2.51]) Consider the DCTRS $\mathcal{R}$ given by*

$$\begin{array}{rcl} g(x) & \to & k(y) \Leftarrow h(x) \to^* d, h(x) \to^* c(y) \\ h(d) & \to & c(a) \\ h(d) & \to & c(b) \\ f(k(a), k(b), x) & \to & f(x, x, x) \end{array}$$

This system is quasi-reductive (and thus cs-quasi-reductive) (cf., [73]). However, the system $U_{cs}(\mathcal{R})$, where the conditional rule is replaced by

$$\begin{array}{rcl} g(x) & \to & U_1(h(x), x) \\ U_1(d, x) & \to & U_2(h(x), x) \\ U_2(c(y), x) & \to & k(y) \end{array}$$

with $\mu(U_i) = \{1\}$ for $i \in \{1, 2\}$, is not μ-terminating.

$$\begin{array}{rl} & f(k(a), k(b), U_2(h(d), d)) \\ \to_{U_{cs}(\mathcal{R})} & f(U_2(h(d), d), U_2(h(d), d), U_2(h(d), d)) \\ \to^+_{U_{cs}(\mathcal{R})} & f(U_2(c(a), d), U_2(c(b), d), U_2(h(d), d)) \\ \to^+_{U_{cs}(\mathcal{R})} & f(k(a), k(b), U_2(h(d), d)) \end{array}$$

Note that in this counterexample the crucial subterm $t' = U_2(h(d), d)$ which reduces to both $k(a)$ and $k(b)$ does not have a counterpart in the original system, i.e., a term $t \in \mathcal{T}(\mathcal{F}, V)$ with $t \to^*_{U_{cs}(\mathcal{R})} t'$. Hence, it seems natural to conjecture that such counterexamples are impossible if we only consider derivations issuing from original terms. This is indeed the case, even for quasi-decreasing systems (cf. Theorems 4 and 5 below).

Definition 7 (μ-termination on original terms)**.** *A CSRS $\mathcal{R} = (U(\mathcal{F}), U(R))$ with replacement map μ, obtained by the transformation of Definition 4 is called* μ-terminating on original terms*, if there is no infinite reduction sequence issuing from a term $t \in \mathcal{T}(\mathcal{F}, V)$ in $\mathcal{R}$.*

Now we can state the main results of this section.

Theorem 4. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. If $\mathcal{R}$ is quasi-decreasing, then $U_{cs}(\mathcal{R})$ is μ-terminating on $\mathcal{T}(\mathcal{F}, V)$.*

Proof. For a proof by minimal counterexample suppose that $s \in \mathcal{T}(\mathcal{F}, V)$ initiates an infinite $\to_{U_{cs}(\mathcal{R})}$-reduction D: $s \to_{U_{cs}(\mathcal{R})} \ldots$ such that there is no $s' \in \mathcal{T}(\mathcal{F}, V)$, $s \succ s'$ with this property (where $\succ$ is the quasi-decreasing ordering). Since $\succ$ contains the subterm ordering, this implies that every proper subterm of s is $\to_{U_{cs}(\mathcal{R})}$-terminating. Hence, D must have at least one root reduction step, i.e., be of the shape $s \to^*_{U_{cs}(\mathcal{R})} t \overset{\epsilon}{\to}_{U_{cs}(\mathcal{R})} u \to_{U_{cs}(\mathcal{R})} \ldots$ where $t \overset{\epsilon}{\to}_{U_{cs}(\mathcal{R})} u$ is the first root reduction step. Since the root symbol of s is from the original signature, the left-hand side of the rule applied to t must be a term of the original signature. There are two possibilities now.

First, assume an unconditional rule $l \to r$ ($l, r \in \mathcal{T}(\mathcal{F}, V)$) was applied to t. Then, $t = l\sigma$, $u = r\sigma$. According to Corollary 2 we have $s \to^*_{U_{cs}(\mathcal{R})} \mathsf{tb}(t) \to^*_{U_{cs}(\mathcal{R})} t$. Since $t = l\sigma$, we get $\mathsf{tb}(t) = l\sigma'$, because the steps from $\mathsf{tb}(t)$ to t are tb-preserving and $x\sigma' \to^*_{U_{cs}(\mathcal{R})} x\sigma$ for all $x \in \mathit{Dom}(\sigma)$. Thus, we have $s \to^*_{U_{cs}(\mathcal{R})} \mathsf{tb}(t) = l\sigma' \to_{U_{cs}(\mathcal{R})} r\sigma' \to^*_{U_{cs}(\mathcal{R})} r\sigma = u$. Furthermore, by quasi-decreasingness we get $s \succ r\sigma'$, because of $\to_{\mathcal{R}} \subseteq \succ$ and $s \to^+_{U_{cs}(\mathcal{R})} r\sigma' \Rightarrow s \to^+_{\mathcal{R}} r\sigma' \in \mathcal{T}(\mathcal{F}, V)$ (according to Theorem 2). This means that in every infinite reduction sequence starting from s we eventually arrive at $r\sigma' \prec s$, which hence also initiates an infinite $\to_{U_{cs}(\mathcal{R})}$-reduction, thus yielding a smaller counterexample (since $s \succ r\sigma'$). But this contradicts our minimality assumption.

Secondly, assume the transformed version of a conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n$ is applied to t. Hence, $t = l\sigma$ and as before we get $\mathsf{tb}(t) = l\sigma'$ where $x\sigma' \to^*_{U_{cs}(\mathcal{R})} x\sigma$ for all $x \in \mathit{Dom}(\sigma)$. Thus $u = U_1(s_1, x_1, \ldots, x_{k_1})\sigma$ and we have $\mathsf{tb}(t) \to_{U_{cs}(\mathcal{R})} U_1(s_1, x_1, \ldots, x_{k_1})\sigma'$. By quasi-decreasingness we get $l\sigma' \succ s_1\sigma', x_1\sigma', \ldots, x_{k_1}\sigma'$, hence all the latter terms are terminating by minimality of the counterexample. Therefore, $s_1\sigma$ and $x_1\sigma, \ldots, x_{k_1}\sigma$ are terminating, too, because of $y\sigma' \to^*_{U_{cs}(\mathcal{R})} y\sigma$ for all $y \in \mathit{Dom}(\sigma)$. Thus, the only possibility of an infinite reduction from u is via a next root reduction step: $u = U_1(s_1, x_1, \ldots, x_{k_1})\sigma \to^*_{U_{cs}(\mathcal{R})} U_1(t_1, x_1, \ldots, x_{k_1})\sigma_1 \xrightarrow{\epsilon}_{U_{cs}(\mathcal{R})} U_2(s_2, x_1, \ldots, x_{k_2})\sigma_1$. So $s_1\sigma' \to^*_{U_{cs}(\mathcal{R})} s_1\sigma \to^*_{U_{cs}(\mathcal{R})} t_1\sigma_1$, and Corollary 2 yields $s_1\sigma' \to^*_{U_{cs}(\mathcal{R})} \mathsf{tb}(t_1\sigma_1) = t_1\sigma'_1 \to^*_{U_{cs}(\mathcal{R})} t_1\sigma_1$. Then it also holds that $U_1(t_1, x_1, \ldots, x_{k_1})\sigma'_1 \to_{U_{cs}(\mathcal{R})} U_2(s_2, x_1, \ldots, x_{k_2})\sigma'_1$ and as $s_1\sigma'_1 \to^*_{U_{cs}(\mathcal{R})} t_1\sigma'_1$, we have $s_1\sigma'_1 \to^*_{\mathcal{R}} t_1\sigma'_1 \in \mathcal{T}(\mathcal{F}, V)$ according to Theorem 2 and thus $l\sigma'_1 \succ s_2\sigma'_1$. By minimality, $s_2\sigma'_1$ and $x_1\sigma'_1, \ldots, x_{k_2}\sigma'_1$ are terminating, hence also $s_2\sigma_1$ and $x_1\sigma_1, \ldots, x_{k_2}\sigma_1$ because of $x\sigma' \to^*_{U_{cs}(\mathcal{R})} x\sigma$ for all $x \in \mathit{Dom}(\sigma)$. Similarly, an infinite reduction from $U_2(s_2, x_1, \ldots, x_{k_2})\sigma_1$ is only possible via a next reduction step for which we need $s_2\sigma_1 \to^*_{U_{cs}(\mathcal{R})} t_2\sigma_2$ for some σ_2. By continuing this argumentation, we finally get that $l\sigma$ must eventually be reduced to $U_n(t_n, x_1, \ldots, x_{k_n})\sigma_n$ and $l\sigma'$ can be reduced to $U(t_n, x_1, \ldots, x_{k_n})\sigma'_n$. We have that $t_n\sigma'_n \in \mathcal{T}(\mathcal{F}, V)$ is terminating by minimality (and quasi-decreasingness) and $t_n\sigma_n$ is terminating because of $t_n\sigma'_n \to^*_{U_{cs}(\mathcal{R})} t_n\sigma_n$. Therefore, the term $U(t_n, x_1, \ldots, x_{k_n})\sigma_n$ is reduced to $r\sigma_n$ and $U(t_n, x_1, \ldots, x_{k_n})\sigma'_n$ can be reduced to $r\sigma'_n$. We have $l\sigma' (= l\sigma'_n) \succ r\sigma'_n$ because of $l\sigma' \to^+_{U_{cs}(\mathcal{R})} r\sigma'_n \in \mathcal{T}(\mathcal{F}, V)$ and thus $l\sigma' \to^+_{\mathcal{R}} r\sigma'_n$ by Theorem 2. Hence, $r\sigma'_n$ (with $s \to^*_{U_{cs}(\mathcal{R})} r\sigma'_n \to^*_{U_{cs}(\mathcal{R})} r\sigma_n$) is terminating because of minimality and $r\sigma_n$ is terminating due to $r\sigma'_n \to^*_{U_{cs}(\mathcal{R})} r\sigma_n$. But this contradicts the counterexample property (of s). Hence, we are done. $\square$

Conversely, cs-quasi-reductivity follows from termination of the transformed system on original terms.

Theorem 5. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. If $U_{cs}(\mathcal{R})$ is μ-terminating on $\mathcal{T}(\mathcal{F}, V)$, then $\mathcal{R}$ is cs-quasi-reductive.*

Proof. We define the ordering $\succ$ by $s \succ t$ if $s \to^{+}_{U_{cs}(\mathcal{R})} t$ and s is reachable (in $\to_{U_{cs}(\mathcal{R})}$) by a term of the original signature (i.e. $\mathsf{tb}(s) \to^{*}_{U_{cs}(\mathcal{R})} s$). This relation is well-founded, because $\to_{U_{cs}(\mathcal{R})}$ is terminating on $\mathcal{T}(\mathcal{F}, V)$. Let $\succ_\mu$ be the μ-monotonic closure of $\succ$ w.r.t. $\mathcal{T}(U(\mathcal{F}), V)$, i.e., $C[s]_p \succ_\mu C[t]_p$ if $s \succ t \wedge p \in Pos^\mu(C[s]_p)$. We show that $\mathcal{R}$ is cs-quasi-reductive w.r.t. $\succ_\mu$. Note that $\succ_\mu \subseteq \to^{+}_{U_{cs}(\mathcal{R})}$.

First, we will deal with well-foundedness of $\succ_\mu$. Consider decreasing $\succ_\mu$-chains starting from a term t. If $s \to^{*}_{U_{cs}(\mathcal{R})} t$ for some term $s \in \mathcal{T}(\mathcal{F}, V)$ (i.e., t is reachable from an original term), there cannot be an infinite decreasing $\succ_\mu$-chain starting from t, because this would contradict termination of $\to_{U_{cs}(\mathcal{R})}$ on $\mathcal{T}(\mathcal{F}, V)$. Otherwise, $t = C[t_1 \dots t_n]_{p_1 \dots p_n}$, such that $s_i \to^{*}_{U_{cs}(\mathcal{R})} t_i$, for some $s_i \in \mathcal{T}(\mathcal{F}, V)$ and $p_i \in Pos^\mu(t)$ for all $i \in \{1, \dots, n\}$ and no proper superterm of any t_i is reachable by a term from $\mathcal{T}(\mathcal{F}, V)$. Thus, if $t \succ_\mu u$, then $u = C[t_1 \dots u_i \dots t_n]_{p_1 \dots p_i \dots p_n}$ and $t_i \succ u_i$. Furthermore, if $u \succ_\mu v$, then $v = C[t_1 \dots u_i \dots v_j \dots t_n]_{p_1 \dots p_i \dots p_j \dots p_n}$ and $t_j \succ v_j$. It is easy to see that there cannot be an infinite decreasing $\succ_\mu$-sequence of this shape, as each decreasing $\succ$-sequence starting at some t_i is finite. Hence, $\succ_\mu$ is well-founded.

If we have $s_i\sigma \succ_\mu t_i\sigma$ for all $1 \leq i < j$, then we get (cf., the proof of Theorem 3) $l\sigma \to^{*}_{U_{cs}(\mathcal{R})} U(s_j, x_1, \dots, x_m)\sigma$ and thus $l\sigma \succ^{st}_{\mu} s_j\sigma$ for all rules $l \to r \Leftarrow s_1 \to^* t_1, \dots, s_n \to^* t_n$, all $0 \leq j \leq n$ and all substitutions $\sigma \colon V \to \mathcal{T}(\mathcal{F}, V)$. Analogously, if $s_i\sigma \succeq_\mu t_i\sigma$ for all $1 \leq i \leq n$, then we have $l\sigma \to^{*}_{U_{cs}(\mathcal{R})} r\sigma$ and thus $l\sigma \succ_\mu r\sigma$.

Hence, $\mathcal{R}$ is cs-quasi-reductive. □

As a corollary we obtain the following equivalences between the various notions.

Corollary 4. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. The following properties of $\mathcal{R}$ are equivalent: μ-termination of $U_{cs}(\mathcal{R})$ on original terms, cs-quasi-reductivity, quasi-decreasingness, and operational termination.*

Proof. The equivalence of quasi-decreasingness and operational termination was proved in [61]. Theorem 5, Proposition 2 and Theorem 4 show: $\mu_{U_{cs}(\mathcal{R})}$-termination of $U_{cs}(\mathcal{R})$ on $\mathcal{T}(\mathcal{F}, V)$ $\Rightarrow$ cs-quasi-reductivity of $\mathcal{R}$ $\Rightarrow$ quasi-decreasingness of $\mathcal{R}$ $\Rightarrow$ $\mu_{U_{cs}(\mathcal{R})}$-termination of $U_{cs}(\mathcal{R})$ on $\mathcal{T}(\mathcal{F}, V)$. □

3.2.3 Disproving Operational Termination

While proving termination on original terms for a concrete CSRS $U_{cs}(\mathcal{R})$ is (at least theoretically) easier than proving general termination of the system (as termination implies termination on original terms, but in general not vice versa), disproving termination on original terms and thus disproving operational termination of the underlying DCTRSs $\mathcal{R}$ might be significantly harder than ordinary non-termination analysis.

In this section we show that the transformation of Definition 4 is complete with respect to collapse-extended termination ($C_{\mathcal{E}}$-termination), thus solving an open problem of [18]. Hence, if a transformed system can be proved to be non-terminating, we can deduce non-$C_{\mathcal{E}}$-operational termination of the underlying DCTRS.

Furthermore, whenever operational termination and $C_{\mathcal{E}}$-operational termination of a DCTRS $\mathcal{R}$ coincide, then $U_{cs}(\mathcal{R})$ is μ-terminating if and only if $\mathcal{R}$ is operationally terminating.

Definition 8 ($C_{\mathcal{E}}$-termination, [36, 73])**.** *We call a CSRS $\mathcal{R}$ with replacement map μ* $C_{\mathcal{E}}$-μ-terminating *(or just $C_{\mathcal{E}}$-terminating) if $\mathcal{R} \uplus C_{\mathcal{E}}$*[4] *with $\mu(G) = \{1,2\}$ is μ-terminating. Moreover, we define $C_{\mathcal{E}} = \{G(x,y) \to x, G(x,y) \to y\}$.*

Definition 9 ($C_{\mathcal{E}}$-cs-quasi-reductivity)**.** *Let $\mathcal{R}$ be a DCTRS. We call $\mathcal{R}$* $C_{\mathcal{E}}$-cs-quasi-reductive *if $\mathcal{R} \uplus C_{\mathcal{E}}$ is cs-quasi-reductive.*

Lemma 4. *Let $U_{cs}(\mathcal{R})$ be a CSRS obtained by the transformation of Definition 4 from a DCTRS $\mathcal{R} = (\mathcal{F}, R)$. If $U_{cs}(\mathcal{R})$ is not μ-terminating, then there exists an infinite reduction sequence starting from a term t, such that $root(t) \in \mathcal{F}$ and every replacing subterm of t is μ-terminating.*

Proof. In the following we call non-μ-terminating terms containing only μ-terminating proper subterms *minimal non-terminating.*

The basic idea of the proof is to show that a minimal non-terminating term u rooted by a U-symbol must either be reduced to a minimal non-terminating term that is not rooted by a U-symbol, or it must contain a (forbidden) U-rooted minimal non-terminating proper subterm. In both cases we will derive a contradiction to the assumption that every minimal non-terminating term is rooted by a U-symbol.

Let $U_1^\alpha, \dots, U_n^\alpha$ be the U-symbols introduced when transforming a conditional rule α (cf. Definition 2). Assume towards a contradiction that

$$U_{cs}(\mathcal{R}) \text{ is not } \mu\text{-terminating and no term } t \text{ as in the lemma exists.} \tag{3.4}$$

Thus, there exists a non-terminating U-term u where every replacing proper subterm of u is μ-terminating, because the existence of a non-μ-terminating term containing only μ-terminating μ-replacing subterms is obvious and this term cannot have a root symbol from $\mathcal{F}$ because of our assumption. Hence, there exists an infinite reduction sequence D starting from u. We inspect D.

Assume $u = U_j^\alpha(u_1, \dots, u_m)$. We first prove the following claim by induction on $n - j$ where n is the number of conditions of α.

> If u is minimal non-terminating, then the forbidden subterm u_i contains an allowed minimal non-terminating subterm for some $2 \leq i \leq m$.

First assume $u = U_n^\alpha(u_1, u_2, \dots, u_m)$ and u is minimal non-terminating. Hence, eventually in every infinite reduction there will be a (first) root reduction step $u \to^*_{U_{cs}(\mathcal{R})} u' \overset{\epsilon}{\to}_{U_{cs}(\mathcal{R})} r\sigma$ where $r \in \mathcal{T}(\mathcal{F}, V)$ (cf. Definition 4). From our assumption

[4]We use the notation $\mathcal{R} \uplus C_{\mathcal{E}}$ as abbreviation for $(\mathcal{F} \uplus \{G\}, R \uplus \{G(x,y) \to x, G(x,y) \to y\})$.

(3.4) it follows that $r\sigma$ must contain a minimal non-terminating U-subterm inside the substitution. The arguments $u_2, \ldots, u_m$ are forbidden for reduction in u, so for every $x \in \mathit{Var}(r)$ either $x\sigma$ occurred as forbidden subterm in u or it occurred allowed in u' in which case it cannot be non-terminating as u' is minimal non-terminating (obviously a minimal non-terminating term cannot be reduced to a term containing a non-terminating proper subterm by reduction steps below the root). Hence, the claim holds.

Second, assume $u = U_j^\alpha(u_1, u_2, \ldots, u_k)$ with $j < n$ and u is minimal non-terminating. In every infinite reduction sequence issuing from u there will be a (first) root reduction step

$$u \rightarrow^*_{U_{cs}(\mathcal{R})} u' \overset{\epsilon}{\rightarrow}_{U_{cs}(\mathcal{R})} u'' = U_{j+1}^\alpha(\_, u_2, \ldots, u_k, u_{k+1}, \ldots, u_{k+l}).$$

The term u'' is non-terminating (as it is part of an infinite reduction) and thus contains an allowed minimal non-terminating subterm. We distinguish two cases

- If u'' itself is minimal non-terminating, then we apply the induction hypothesis yielding that an allowed subterm of u_i is minimal non-terminating for some $i \in \{2, \ldots k+l\}$. The terms $\{u_{k+1}, \ldots, u_{k+l}\}$ occurred at allowed positions in u' (these terms are variable bindings of variables occurring in the right-hand side of the j^{th} condition of α). Thus they cannot contain a minimal non-terminating allowed subterm as this would contradict minimal non-termination of u' and thus of u. Hence, one of the terms $u_2, \ldots u_k$ contains an allowed minimal non-terminating subterm.

- If a proper subterm of u'' is minimal non-terminating, then this subterm must be in the substitution part of $r\sigma = U_{j+1}^\alpha(s, x_2, \ldots x_{k+l})\sigma = u''$, where r is the right-hand side of the rule applied in the root reduction, because all proper subterms of r are either variables or rooted by symbols from $\mathcal{F}$ and thus cannot be minimal non-terminating because of assumption (3.4). However, for every $x \in \mathit{Var}(r)$, the term $x\sigma$ already occurred in u' and as u' is minimal non-terminating, the terms $x_{k+1}\sigma, \ldots, x_{k+l}\sigma$ are terminating. Hence, an allowed subterm of $x_i\sigma$ is minimal non-terminating for some $i \in \{2, \ldots, k\}$.

Now we have shown that under assumption (3.4) it holds that every minimal non-terminating term contains a forbidden (and thus proper) subterm with the same property which is obviously a contradiction. Hence, assumption (3.4) cannot hold and the lemma is proved. □

The following definition will be useful in proving the subsequent completeness result concerning termination.

Definition 10 (partial evaluation). *Let $U_{cs}(\mathcal{R})$ be a CSRS obtained from a DCTRS $\mathcal{R} = (\mathcal{F}, R)$ by the transformation of Definition 4 and let t be a term such that every* *maximal U-rooted subterm of t is μ-terminating (w.r.t. $U_{cs}(\mathcal{R})$). Then we define $peval_{\mathcal{R}}(t)$ as*

$$peval_{\mathcal{R}}(t) = \begin{cases} x, & \text{if } t = x \in V \\ f(peval_{\mathcal{R}}(v_1), \dots, peval_{\mathcal{R}}(v_n)), & \text{if } t = f(v_1, \dots, v_n) \text{ and } f \in \mathcal{F} \\ G'(peval_{\mathcal{R}}(u_1), \dots, peval_{\mathcal{R}}(u_m)), & \text{if } t = U_i^{\alpha}(v_1, \dots, v_n) \text{ and } U_i^{\alpha} \notin \mathcal{F} \end{cases}$$

where $G'(g_1, \dots, g_k)$ stands for $G(g_1, G(g_2, \dots G(g_{k-1}, G(g_k, A)) \dots))$ or A and

$$\{u_1, \dots, u_m\} = \{u \mid t \to^+_{U_{cs}(\mathcal{R})} u, root(u) \in \mathcal{F} \cup Var\}.$$

Moreover, the order of the terms $u_1, \dots, u_m$ is determined by an arbitrary but fixed order on terms. If there is no such term, then $peval(t) = A$. Here, A is a fresh constant and G is a fresh binary symbol (which will be used as non-deterministic projection symbol, i.e., by including the rules $G(x, y) \to x$, $G(x, y) \to y$, in Theorem 6 below).

Note that whenever a term t is μ-terminating and $t \to^*_{U_{cs}(\mathcal{R})} t'$, then the maximal U-rooted subterms of t' are μ-terminating as well. This is because maximality of the U-rooted subterms means that all function symbols occurring above these subterms are original function symbols. Hence, as all arguments of all original function symbols are replacing, the maximal U-rooted subterms are replacing. So non-μ-termination of the maximal U-rooted subterms of t' would imply non-μ-termination of t' and thus non-μ-termination of t. Thus, as t is μ-terminating, the maximal U-rooted subterms of t' are μ-terminating as well. Hence, *peval* is well-defined.

Informally, *peval*(t) represents all descendants of t (w.r.t. $\to_{U_{cs}(\mathcal{R})}$) that do not contain any U-symbols.

Definition 11 (correspondence w.r.t to *peval*)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS and $U_{cs}(\mathcal{R})$ be the system obtained by the transformation of Definition 4. Furthermore, let $s, t \in \mathcal{T}(U(\mathcal{F}) \uplus \{G, A\}, V)$. We say that s weakly corresponds to t w.r.t. peval, denoted by $t \curvearrowright s$, if $s = C[s_1, \dots, s_n]_{p_1,\dots,p_n}$, $t = C[t_1, \dots, t_n]_{p_1,\dots,p_n}$, and for all $1 \leq i \leq n$ we have that t_i is a μ-terminating U-term with $s_i = peval(t_i)$.*

Note that the context C in Definition 11 may contain U-symbols and is unique for all terms t and s with $t \curvearrowright s$, because $root(s_i) \in \mathcal{F} \uplus \{G, A\}$ and $root(t_i) \in U(\mathcal{F}) \setminus \mathcal{F}$ and thus $s_i \neq t_i$ for all $1 \leq i \leq n$. Hence, C is the *maximal* context such that $s = C[s_1, \dots, s_n]_{p_1,\dots,p_n}$ and $t = C[t_1, \dots, t_n]_{p_1,\dots,p_n}$. However, note that the t_i's are not necessarily the maximal U-rooted subterms of t.

Lemma 5. *Let $\mathcal{R}$ be a DCTRS and let $U_{cs}(\mathcal{R})$ be the system obtained by the transformation of Definition 4. Given two terms $s, t \in \mathcal{T}(U(\mathcal{F}) \cup \{G, A\}, V)$ with $t \curvearrowright s$, i.e. $t = C[t_1, \dots, t_n]_{p_1,\dots,p_n}$ and $s = C[peval(t_1), \dots, peval(t_n)]_{p_1,\dots,p_n}$*

1. *$t \xrightarrow{q}_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} t'$ and $q \geq p_i$ for some $1 \leq i \leq n$ implies $s \to^*_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} s'$ and $t' \curvearrowright s'$, and*

2. $t \xrightarrow{q}_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} t'$ *and* $q < p_i$ *for some* $1 \leq i \leq n$ *implies* $s \to^{+}_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} s'$ *and* $t' \curvearrowright s'$.

Proof. (1.) Let $q \geq p_j$. The term $peval(t'|_{p_j}) = G'(peval(u_1), \ldots, peval(u_n))$ where the set $\{u_1, \ldots, u_n\}$ is the set of all terms u_i satisfying that $t'|_{p_j} \to^{*}_{U_{cs}(\mathcal{R})} u_i$ and $root(u_i) \in \mathcal{F} \cup Var$ according to Definition 10.

On the other hand $peval(t|_{p_j}) = G'(peval(v_1), \ldots, peval(v_m))$. Whenever we have that $t'|_{p_j} \to^{*}_{U_{cs}(\mathcal{R})} u_i$, then also $t|_{p_j} \to^{*}_{U_{cs}(\mathcal{R})} u_i$, because $t|_{p_j} \to_{U_{cs}(\mathcal{R})} t'|_{p_j} \to^{*}_{U_{cs}(\mathcal{R})} u_i$. Hence, $\{u_1, \ldots, u_n\} \subseteq \{v_1, \ldots, v_m\}$ and $peval(t|_{p_j}) \to^{*}_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} peval(t'|_{p_j})$ by applying G-rules to filter those v_is that do not occur in $\{u_1, \ldots, u_n\}$. Hence, $s = s[peval(t|_{p_j})]_{p_j} \to^{*}_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} s[peval(t'|_{p_j})]_{p_j} = s'$.

(2.) Let $q < p_i$ for some $1 \leq i \leq n$. We have $t = t[l\sigma]_q$ and thus $s = s[l\sigma']_q$, because $q < p_i$ for some i, and hence $q.q' < p_i$ for all $q' \in Pos_{U(\mathcal{F}) \cup \{G\}}(l)$, because l does not contain a U-symbol below the root and $t|_{p_i}$ is a U-term for all i. Moreover, for all $1 \leq i, j \leq n$ we have that $t_i = t_j$ implies $peval(t_i) = peval(t_j)$. Hence, l matches $s|_q$ even if it is non-linear. Obviously, $x\sigma \curvearrowright x\sigma'$ for all $x \in Dom(\sigma)$, because $x\sigma$ cannot be a proper subterm of $t|_{p_i}$ for any i.

Hence, we have $s = s[l\sigma']_q \to_{U_{cs}(\mathcal{R}) \cup C_{\mathcal{E}}} s' = s[r\sigma']_q$ and $t' = t[r\sigma]_q \curvearrowright s'$, because $s' = C'[peval(t'_1), \ldots peval(t'_m)]_{q_1,\ldots,q_m}$ and $t' = C'[t'_1, \ldots t'_m]_{q_1,\ldots,q_m}$ where t'_i is μ-terminating for all $1 \leq i \leq m$, since it is equal to t_j for some $1 \leq j \leq n$. □

Theorem 6 (completeness for $C_{\mathcal{E}}$-termination)**.** *Let $\mathcal{R}$ be a DCTRS and let $U_{cs}(\mathcal{R})$ be its transformed system according to Definition 4. Then $\mathcal{R}$ is $C_{\mathcal{E}}$-cs-quasi-reductive if and only if $U_{cs}(\mathcal{R})$ is $C_{\mathcal{E}}$-μ-terminating.*

Proof. $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}}) = U_{cs}(\mathcal{R}) \uplus C_{\mathcal{E}}$ and $\mathcal{R}^{C_{\mathcal{E}}} = \mathcal{R} \uplus C_{\mathcal{E}}$. Note that $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})$ is the system obtained by transforming $\mathcal{R}^{C_{\mathcal{E}}}$.

The *if* part of the proof is therefore covered by Theorem 5, because μ-termination of $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})$ implies cs-quasi-reductivity of $\mathcal{R}^{C_{\mathcal{E}}}$.

The *only if* part of the theorem will be proved indirectly by showing that non-μ-termination of $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})$ implies non-μ-termination of $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})$ on original terms, i.e. terms of the original signature of $\mathcal{R}$ (plus $\{G, A\}$), which further implies non-cs-quasi-reductivity of $\mathcal{R}^{C_{\mathcal{E}}}$ according to Theorem 4.

So assume $U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})$ is non-terminating. According to Lemma 4 there exists an infinite reduction sequence $D\colon t_0 \to^{*}_{U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})} t_1 \to^{*}_{U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})} \cdots$ starting from a term t_0 with a root symbol from $\mathcal{F} \uplus \{G, A\}$, such that each replacing subterm of t_0 is terminating. We will prove the existence of another infinite reduction D' starting at $t'_0 = peval_{\mathcal{R}^{C_{\mathcal{E}}}}(t_0)$, which does not contain any U-symbols. Note that $t_0 = C[t_0^1, \ldots t_0^m]_{p_1,\ldots,p_m} \curvearrowright t'_0 = C[peval(t_0^1), \ldots peval(t_0^m)]_{p_1,\ldots,p_m}$ where C is non-empty, because t_0 is not a U-term.

Now to prove by induction that an infinite reduction sequence D' starting at t'_0 can be constructed we show that $t_j \curvearrowright t'_j$ implies $t_{j+k} \curvearrowright t'_{j+k}$ for some $k \geq 1$ with $t'_j \to^{+}_{U_{cs}(\mathcal{R}^{C_{\mathcal{E}}})} t'_{j+k}$.

Assume $t_j \curvearrowright t'_j$, i.e. let $t_j = C[t^1_j, \ldots, t^n_j]_{p_1,\ldots,p_n}$ and let $t'_j = C[peval(t^1_j), \ldots, peval(t^n_j)]_{p_1,\ldots,p_n}$. Consider the subreduction $t_j \rightarrow_{U_{cs}(\mathcal{R}^{C_\mathcal{E}})} t_{j+1} \cdots \rightarrow_{U_{cs}(\mathcal{R}^{C_\mathcal{E}})} t_{j+k}$ of D such that the last step of this subreduction occurs at a position $q < p_i$ for some $1 \leq i \leq n$. Note that such a reduction must appear in each tail of D, because the terms $t^1_j, \ldots t^n_j$ are all μ-terminating.

We get $t'_j \rightarrow^+_{U_{cs}(\mathcal{R}^{C_\mathcal{E}})} t'_{j+k}$ and $t_{j+k} \curvearrowright t'_{j+k}$ through iterated (k times) applications of Lemma 5.

Hence, we can construct an infinite $U_{cs}(\mathcal{R}^{C_\mathcal{E}})$-reduction sequence starting from t'_0 which implies non-cs-quasi-reductivity of $\mathcal{R}^{C_\mathcal{E}}$ according to Corollary 4. □

As corollaries of Theorem 6 we get the following modularity results.

Corollary 5. *Let $\mathcal{R}$ and $\mathcal{S}$ be DCTRSs with disjoint signatures that are both $C_\mathcal{E}$-cs-quasi-reductive. Then $\mathcal{R} \uplus \mathcal{S}$ is $C_\mathcal{E}$-cs-quasi-reductive.*

Proof. According to Theorem 6, $U_{cs}(\mathcal{R})$ and $U_{cs}(\mathcal{S})$ are $C_\mathcal{E}$-μ-terminating. In [36], modularity of $C_\mathcal{E}$-μ-termination is proved. Thus, $U_{cs}(\mathcal{R}) \uplus U_{cs}(\mathcal{S})$ is $C_\mathcal{E}$-μ-terminating. As $U_{cs}(\mathcal{R}) \uplus U_{cs}(\mathcal{S}) = U_{cs}(\mathcal{R} \uplus \mathcal{S})$, $\mathcal{R} \uplus \mathcal{S}$ is $C_\mathcal{E}$-cs-quasi-reductive. □

Corollary 6. *Let $\mathcal{R}$ and $\mathcal{S}$ be DCTRSs with disjoint signatures that are both $C_\mathcal{E}$-operationally terminating (which means for a DCTRS $\mathcal{R}$ that $\mathcal{R} \uplus C_\mathcal{E}$ is operationally terminating). Then $\mathcal{R} \uplus \mathcal{S}$ is $C_\mathcal{E}$-operationally terminating.*

Example 22. *Consider the following DCTRS $\mathcal{R}$*

$$\begin{array}{rcl}
\alpha_1\colon div(x,y) \rightarrow pair(0,x) & \Leftarrow & greater(y,x) \rightarrow^* true \\
\alpha_2\colon div(x,y) \rightarrow pair(s(q),r) & \Leftarrow & leq(y,x) \rightarrow^* true, \\
 & & div(x-y,y) \rightarrow^* pair(q,r) \\
x - 0 & \rightarrow & x \\
0 - y & \rightarrow & 0 \\
s(x) - s(y) & \rightarrow & x - y \\
greater(s(x),s(y)) & \rightarrow & greater(x,y) \\
greater(s(x),0) & \rightarrow & true \\
leq(s(x),s(y)) & \rightarrow & leq(x,y) \\
leq(0,x) & \rightarrow & true
\end{array}$$

performing a simple division with remainder. Transforming the conditional rules α_1 and α_2 yields

$$\begin{array}{rcl}
div(x,y) & \rightarrow & U^{\alpha_1}_1(greater(y,x),x,y) \\
U^{\alpha_1}_1(true,x,y) & \rightarrow & pair(0,x) \\
div(x,y) & \rightarrow & U^{\alpha_2}_1(leq(y,x),x,y) \\
U^{\alpha_2}_1(true,x,y) & \rightarrow & U^{\alpha_2}_2(div(x-y,y),x,y) \\
U^{\alpha_2}_2(pair(q,r),x,y) & \rightarrow & pair(s(q),r)
\end{array}$$

Property of $U_{cs}(\mathcal{R})$	**Implied property of $\mathcal{R}$**	**Proved in**
μ-Termination	Operational termination	Theorem 3 and Corollary 1
Non-μ-termination	Non-($C_{\mathcal{E}}$-operational termination)	Theorem 6
μ-Termination on original terms	Operational termination	Theorem 5 and Corollary 1
Non-μ-(termination on original terms)	Non-(operational termination)	Theorem 4
$C_{\mathcal{E}}$-termination	$C_{\mathcal{E}}$-operational termination	Theorem 6
Non-($C_{\mathcal{E}}$-termination)	Non-($C_{\mathcal{E}}$-operational termination)	Theorem 6

Table 3.1: Properties of $U_{cs}(\mathcal{R})$ and the implied properties of a DCTRS $\mathcal{R}$.

$U_{cs}(\mathcal{R})$ consists of these rules and the unconditional rules from $\mathcal{R}$. Indeed $U_{cs}(\mathcal{R})$ is non-μ-terminating

$$\begin{array}{rcl} \underline{div(x,0)} & \to & U_1^{\alpha_2}(\underline{leq(0,x)},x,0) \to \underline{U_1^{\alpha_2}(true,x,0)} \\ & \to & U_2^{\alpha_2}(div(\underline{minus(x,0)},0),x,0) \to U_2^{\alpha_2}(\underline{div(x,0)},x,0) \to \dots \end{array}$$

Hence, we deduce non-$C_{\mathcal{E}}$-operational termination of $\mathcal{R}$ according to Theorem 6 which points to a flaw in the specification of $\mathcal{R}$ allowing division by zero.

Table 3.1 summarizes the relations between a DCTRS $\mathcal{R}$ and $U_{cs}(\mathcal{R})$.

3.2.4 Proving Termination on Original Terms Automatically

Theorem 5 suggests that in order to prove operational termination of a DCTRS $\mathcal{R}$, termination of $U_{cs}(\mathcal{R})$ on original terms has to be proved. However, although termination on original terms is a weaker property than ordinary termination, its analysis might be harder and has, despite being an interesting problem, to the author's knowledge, rarely been investigated (with the notable exception of [26]).

In the following, we introduce a simple approach to deal with this problem based on the dependency pair framework of [33, 2]. We refer to the property of a CSRS $((\mathcal{F},R),\mu)$ being μ-terminating on a set of terms identified by a subsignature $\mathcal{F}'$ of $\mathcal{F}$ as *($\mathcal{F}'$)-subsignature termination* or just subsignature termination if $\mathcal{F}'$ is clear from the context.

In our setting we extend the notion of dependency pair problems, in order to take into account our intention of proving termination only on restricted sets of terms, by adding an additional component specifying a (sub-)signature. Thus, we define SS-CS-DP problems (*subsignature context-sensitive dependency pair problems*) to be *quadruples* $(DP,\mathcal{R},\mu,\mathcal{F}')$ where $DP = (\overline{\mathcal{F}},\overline{R})$ and $\mathcal{R} = (\mathcal{F},R)$ are TRSs, μ is a replacement map for the combined signature $\overline{\mathcal{F}} \cup \mathcal{F}$, and $\mathcal{F}' \subseteq \mathcal{F}$ is a signature determining the starting terms, whose μ-termination we are interested in. An SS-CS-DP problem $(DP, \mathcal{R}, \mu, \mathcal{F}')$ is *finite* if there is no infinite $(DP, \mathcal{R}, \mu)$-chain $u_1 \to v_1, u_2 \to v_2, \ldots$ using a substitution σ such that every proper subterm of $u_i\sigma$ and every proper subterm of $v_i\sigma$ is a term from $\mathcal{T}(\mathcal{F}, V)$ and $\mathcal{R}$-reachable by some term from $\mathcal{T}(\mathcal{F}', V)$. An SS-CS-DP problem is *infinite* if it is not finite. Note that we do not restrict the root symbols of dependency pairs to dependency pair symbols originating from symbols of $\mathcal{F}'$. While such a restriction would be needed for a general approach to prove termination on a restricted set of terms with dependency pairs, it is not needed for the particular case of CSRSs that are transformed DCTRSs obtained by the transformation of Definition 4 (cf. the ONLY IF part of the proof of Theorem 7 below).[5]

Analogously to the case without subsignature restriction dealt with in [2, Theorem 12], we can characterize termination of a CSRS on terms identified by a subsignature by finiteness of a corresponding SS-CS-DP-problem.

Theorem 7. *Let $\mathcal{R}' = (\mathcal{F}', R)$ be a DCTRS and let $\mathcal{R} = U_{cs}(\mathcal{R}')$. $\mathcal{R}$ is μ-terminating on terms $\mathcal{T}(\mathcal{F}', R)$ iff the SS-CS-DP-problem $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu, \mathcal{F}')$ is finite.*

Proof. IF: Assume $\mathcal{R}$ is not μ-terminating on terms of $\mathcal{T}(\mathcal{F}', V)$. Then there exists an infinite sequence of terms $t_1, t_1', s_1, t_2, t_2', s_2, \ldots$ with

$$t_1 \overset{>\epsilon}{\to}{}^*_{\mathcal{R},\mu} t_1' \overset{\epsilon}{\to}_{\mathcal{R},\mu} s_1 \trianglerighteq_\mu t_2 \overset{>\epsilon}{\to}{}^*_{\mathcal{R},\mu} t_2' \overset{\epsilon}{\to}_{\mathcal{R},\mu} s_2 \trianglerighteq_\mu t_3 \overset{>\epsilon}{\to}{}^*_{\mathcal{R},\mu} t_3' \overset{\epsilon}{\to}_{\mathcal{R},\mu} \ldots$$

such that $t_1 \in \mathcal{T}(\mathcal{F}', V)$ and t_i and t_i' are minimal non-μ-terminating for all i, i.e., there is an infinite reduction sequence starting from t_i (resp. t_i'), but all their proper replacing subterms are μ-terminating.

According to the proof of [2, Theorem 12] there exists also a $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu)$-chain $u_1 \to v_1, u_2 \to v_2, \ldots$ enabled by a substitution σ where every proper subterm of $u_i\sigma$ and $v_i\sigma$ appears as subterm of t_j or t_j' or s_j for all $i \geq 1$ and some $j \geq 1$. Since t_i, s_i and t_i' are either reachable from t_1 or are (replacing) subterms of terms reachable from t_1, by Lemma 2 and Definition 4 it follows that all proper subterms of t_j, s_j and t_j' are reachable from terms from $\mathcal{T}(\mathcal{F}', V)$ and hence all proper subterms of $u_i\sigma$ and $v_i\sigma$ are reachable by terms from $\mathcal{T}(\mathcal{F}', V)$ for all $i, j \geq 1$.

ONLY IF: Assume there is an infinite $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu)$-chain $S\colon u_1 \to v_1, u_2 \to v_2, \ldots$ enabled by the substitution σ, such that every proper subterm of $u_i\sigma$ and $v_i\sigma$ is $\mathcal{R}$-reachable from a term of $\mathcal{T}(\mathcal{F}', V)$ for all $i \geq 1$. Note that every suffix of this chain is a $(DP(\mathcal{R}, \mu), \mathcal{R}, \mu)$-chain with the same property as well.

We first prove that w.l.o.g. we can assume that $root(u_1) = f^\#$ for some $f \in \mathcal{F}'$. If S contains some DP $u \to v$ with $root(u) = f^\#$ and $f \in \mathcal{F}'$ we can just use the suffix

[5]Our definition of finiteness of SS-CS-DP problems and Theorem 7 below slightly differ from [81]. There, the definition of (finiteness of) SS-CS-DP problems is partially misleading and [81][Proposition 5] is incorrect for general CSRSs and subsignatures.

of S starting at $u \to v$. Second, assume S contains a DP $u \to v$ with $root(u) = D^\#$ (where $D^\#$ is the symbol introduced during the creation of the dependency pairs (cf. Section 2.4)). From the proof of [2, Theorem 12] we learn that not every DP on S can have a lhs rooted by $D^\#$. Hence eventually S must contain a DP $u' \to v'$ with $root(u') = D^\#$ and $root(v') \neq D^\#$. However, this implies that $root(v') = f^\#$ for some $f \in \mathcal{F}'$, because no U-term is hidden in the rules of $\mathcal{R}$ (cf. the Definition of $DP(\mathcal{R}, \mu)$ in Section 2.4). The lhs of the DP immediately succeeding $u' \to v'$ has the same root as v' and thus we can use the suffix of S starting at that DP.

The only possibility left is the one where the lhs's of all DPs in S are rooted by symbols $f^\#$ with f being a U-symbol. By the proof of [2, Theorem 12] we can construct an infinite reduction sequence in $(\mathcal{R}, \mu)$ out of an infinite DP-chain in such a way that every single term is U-rooted and such that there are infinitely many root reduction steps. However, it is easy to see that such a reduction sequence cannot exist in $\mathcal{R}$. Hence, we get a contradiction.

Thus, we can assume that $root(u_1) = f^\#$ for some $f \in \mathcal{F}'$. Now consider the infinite context-sensitive $\mathcal{R}$-reduction sequence corresponding to our infinite DP chain proved to exist in the proof of [2, Theorem 12]. This reduction sequence starts with a term $f(t_1, \ldots, t_m)$. We have $f \in \mathcal{F}'$ and t_i is reachable from a term of $\mathcal{T}(\mathcal{F}', V)$ for every $1 \leq i \leq m$. Hence, $f(t_1, \ldots, t_m)$ is reachable from such a term and we obtain an infinite reduction sequence in $\mathcal{R}$ starting from a term of $\mathcal{T}(\mathcal{F}', V)$. □

Following the dependency pair framework of [33], an *SS-CS-dependency pair processor* (SS-CS-DP-processor) is a function *Proc* that takes as input an SS-CS-DP-problem and returns either a set of SS-CS-DP problems or "`no`". We call an SS-CS-DP-processor *sound* if finiteness of all SS-CS-DP-problems in $Proc(d)$ implies finiteness of the input SS-CS-DP-problem d. An SS-CS-DP-processor is *complete* if for all SS-CS-DP-problems d, d is infinite whenever $Proc(d)$ is "`no`" or $Proc(d)$ contains an infinite SS-CS-DP-problem.

Since SS-CS-DP-problems $(DP, \mathcal{R}, \mu, \mathcal{F}')$ are finite if there are no $(DP, \mathcal{R}, \mu)$-chains of a certain kind, sound processors for the context-sensitive DP framework of [2] that do not modify but at most eliminate dependency pairs, such as reduction pair processors or the dependency graph processors are sound also for SS-CS-DP-problems in the following sense: If $(DP, \mathcal{R}, \mu, \mathcal{F}')$ is an infinite SS-CS-DP-problem and $Proc((DP, \mathcal{R}, \mu)) = \{(DP_1, \mathcal{R}, \mu), \ldots, (DP_n, \mathcal{R}, \mu)\}$ for a reduction pair or dependency graph processor *Proc* that is sound for the context-sensitive DP framework of [2], then at least one of the SS-CS-DP-problems in the set $\{(DP_1, \mathcal{R}, \mu, \mathcal{F}'), \ldots, (DP_n, \mathcal{R}, \mu, \mathcal{F}')\}$ is infinite. However, by using these processors one ignores the subsignature component of SS-CS-DP-problems. In order to take this subsignature into account in proofs of termination, we present several dedicated SS-CS-DP processors.

3.2.4.1 Narrowing Processors

First, we introduce two SS-CS-DP-processors that build upon the well-known narrowing processor for the dependency pair framework (see e.g. [33]).

The basic idea of the narrowing processor is to anticipate the first step of all possible rewrite sequences in a potential dependency pair chain between two dependency pairs. If $s_i\sigma \to^* t_{i+1}\sigma$ is part of a chain and $s_i\sigma$ and $t_{i+1}\sigma$ are not equal (actually we demand that s_i and t_{i+1} are not unifiable), then the rewrite sequence $s_i\sigma \to^* t_{i+1}\sigma$ is non-empty and contains at least one reduction step at a position $p \in Pos_{\mathcal{F}}(s_i)$ (see the proof of Theorem 8 below for a justification of this claim). Thus, all possibilities of the first such step are covered by replacing $t_i \to s_i$ by the set $\{t_i\theta_j \to s_i^j \mid 1 \leq j \leq n\}$ with $s_i^1, \dots s_i^n$ being all possible (one-step, context-sensitive) narrowings of s_i and $\theta_1, \dots, \theta_n$ being the corresponding mgu's. Theorem 8 below shows that replacing a rule $t_i \to s_i \in DP$ in an SS-CS-DP-problem $\mathcal{P} = (DP, \mathcal{R}, \mu, \mathcal{F}')$ by the set of narrowings does neither alter finiteness nor infinity of $\mathcal{P}$ provided that s_i is linear and does not unify with a left-hand side of any rule in DP.

Analogously, a rule $t_i \to s_i$ occurring in a chain can be replaced under the corresponding preconditions by the set $\{t_i^j \to s_i\theta_j \mid 1 \leq j \leq m\}$, where $t_i^1, \dots t_i^m$ are the (one-step, context-sensitive) *backward* narrowings of t_i and $\theta_1, \dots, \theta_m$ are the corresponding mgu's.

Applying these narrowing approaches in proofs of termination of CSRSs, obtained from DCTRSs by the transformation of Definition 4, allows us to restrict the set of narrowings that we have to consider.

The following lemma provides the basis for this restriction. It provides sufficient conditions for the existence of infinite DP chains enabled by substitutions ranging only into terms over the subsignature of an SS-CS-DP problem.

Lemma 6. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a DCTRS. Assume that $u_1 \to v_1, u_2 \to v_2 \dots$ is an infinite $(\mathcal{P}, U_{cs}(\mathcal{R}), \mu)$-chain enabled by σ where $\mathcal{P} = (\mathcal{F}^{\#}, P)$ such that $\mathcal{F}^\sharp \cap (U(\mathcal{F}) \setminus \mathcal{F}) = \emptyset$ and every proper subterm of $u_i\sigma$ and $v_i\sigma$ is reachable from a term of $\mathcal{T}(\mathcal{F}, V)$ for all $i \geq 1$. Then there is also an infinite $(\mathcal{P}, U_{cs}(\mathcal{R}), \mu)$-chain $u_1' \to v_1', u_2' \to v_2'$ enabled by σ', such that $u_i'\sigma'$ and $v_i'\sigma'$ are terms from $\mathcal{T}(\mathcal{F}^\sharp \cap (U(\mathcal{F}) \setminus \mathcal{F}), V)$ for all $i \geq 1$.*

Proof. Consider the original DP chain $u_1 \to v_1, u_2 \to v_2 \dots$ enabled by σ. According to Theorems 2 and 1 we obtain $v_i\, tb(\sigma) \to^*_{U_{cs}(\mathcal{R})} u_{i+1}\, tb(\sigma)$ where $tb(\sigma)$ is given by $x\, tb(\sigma) = tb(x\sigma)$ for all $x \in Dom(\sigma)$. Hence, $tb(\sigma)$ enables the same DP chain and every term $u_i\, tb(\sigma)$ and $v_i\, tb(\sigma)$ is from $\mathcal{T}(\mathcal{F}^\sharp \cap (U(\mathcal{F}) \setminus \mathcal{F}), V)$ for all $i \geq 1$. $\square$

Lemma 6 motivates the definition of two dependency pair processors based on the standard narrowing processor.

Definition 12 (restricted forward narrowing)**.** *Let $(DP, \mathcal{R}, \mu, \mathcal{F}')$ be an SS-CS-DP-problem with $\mathcal{R} = (\mathcal{F}, R)$. If $u_i \to v_i \in DP$, $\overline{Var^{\mu}}(u_i) \cap Var^{\mu}(v_i) = \emptyset$, v_i is not* *unifiable with any left-hand side of a rule in DP and v_i is linear, then $Proc_{\text{RFN}}$ yields a new SS-CS-DP-problem $(DP', \mathcal{R}, \mu, \mathcal{F}')$ where*

$$DP' = (DP - \{u_i \to v_i\}) \cup \{u_i^k\theta_k \to v_i^k \mid 1 \leq k \leq n\}$$

and $\{v_i^1, \ldots, v_i^n\}$ is the set of all (one-step, context-sensitive) narrowings of v_i with corresponding mgu's $\theta_1, \ldots, \theta_n$, such that all subterms of v_i^k are reducible *to $\mathcal{F}'$-terms for all $1 \leq k \leq n$.*

Theorem 8. *The dependency pair processor $Proc_{\text{RFN}}$ is sound and complete for an SS-CS-DP-problem $P = (\mathcal{P}, \mathcal{R}, \mu, \mathcal{F}')$ where $\mathcal{P} = (\mathcal{F}^\sharp, S^\sharp)$ and $\mathcal{R} = (\mathcal{F}, R)$ provided that $(\mathcal{R}, \mu')$ is obtained by the transformation of Definition 4 from some DCTRS $\mathcal{R}'$ ($\mu'(f) = \mu(f)$ for all $f \in Dom(\mu')$) and $\mathcal{F}^\sharp \cap (\mathcal{F} \setminus \mathcal{F}') = \emptyset$ (i.e., $\mathcal{F}^\sharp$ does not contain any U-symbols).*

Proof. Soundness: Lemma 6 shows that if P is infinite, then there exists an infinite dependency pair chain $v_1 \to u_1, \ldots, v_i \to u_i, s \to t, v_{i+1} \to u_{i+1}, \ldots$ enabled by a substitution θ such that $v_i\theta$, $s\theta$, $t\theta$ and $u_i\theta$ contain no symbols from $\mathcal{F} \setminus \mathcal{F}'$ for all $i \geq 1$. We first show that we can assume w.l.o.g. that no term in the reduction sequences $u_k\theta \to_{\mathcal{R}}^* v_{k+1}\theta$ (resp. $u_i\theta \to_{\mathcal{R}}^* s\theta$ and $t\theta \to_{\mathcal{R}}^* v_{i+1}\theta$) contains a U-term not reducible to an original term (for all $k \geq 1, k \neq i$): By Theorem 2 we get $u_k\theta \to_{\mathcal{R}'}^* v_{k+1}\theta$ for the DCTRS $\mathcal{R}'$ from which $\mathcal{R}$ originated (where the signature of $\mathcal{R}'$ is enriched by the symbols of $\mathcal{F}^\#$). Now following the constructive method of simulating $\mathcal{R}'$-reductions in the proof of Theorem 1 no U-terms that are not reducible to terms over $\mathcal{F}'$ are introduced in the reduction sequence $u_k\theta \to_{\mathcal{R}}^* v_{k+1}\theta$ for all $k \geq 1, k \neq i$.

Thus, there exist substitutions θ enabling the above DP chain such that no terms in reductions $u_k\theta \to_{\mathcal{R}}^* v_{k+1}\theta$ (resp. $u_i\theta \to_{\mathcal{R}}^* s\theta$ and $t\theta \to_{\mathcal{R}}^* v_{i+1}\theta$) contain U-subterms not reducible to terms over $\mathcal{F}'$. Let S be the set of all these substitutions (note that substitutions from S may replace variables by terms containing U-terms, but only ones that are reducible to terms over $\mathcal{F}'$). Moreover, let $\sigma \in S$ be the substitution such that the reduction sequence $t\sigma \to_{U_{cs}(\mathcal{R})}^* v_{i+1}\sigma$ has minimal length (among all substitutions in S).

We take a closer look at the sequence $t\sigma \to_{U_{cs}(\mathcal{R})}^* v_{i+1}\sigma$ and show that due to the minimality of its length the first reduction step must take place at a position $p \in Pos_{\mathcal{F}}(t)$: Assume that the first step is at position $q \notin Pos_{\mathcal{F}}(t)$ and $t|_q = x$. Thus

$$t\sigma \xrightarrow{q} t' = t\sigma' \to^* v_{i+1}\sigma$$

We define a new substitution σ' by $x\sigma' = t'|_q$ and $y\sigma' = y\sigma$ for all $y \neq x$. Since all pairs in a chain are considered to be variable disjoint, we have $u_i\sigma' = u_i\sigma \to_{U_{cs}(\mathcal{R})}^* s\sigma \to_{U_{cs}(\mathcal{R})} s\sigma'$, $t\sigma' \to_{U_{cs}(\mathcal{R})}^* v_{i+1}\sigma'$ and $v_j\sigma' \to_{U_{cs}(\mathcal{R})}^* u_{j+1}\sigma'$ for all $\{j > 0 \mid j \neq i\}$. Thus, the reduction sequence $t\sigma' \to_{U_{cs}(\mathcal{R})}^* v_{i+1}\sigma'$ has a smaller length than $t\sigma \to_{U_{cs}(\mathcal{R})}^* v_{i+1}\sigma$ which contradicts our minimality assumption for σ. Note that the existence of the subsequence $s\sigma \to_{U_{cs}(\mathcal{R})} s\sigma'$ is guaranteed by the fact that $\overline{Var^\mu}(s) \cap Var^\mu(t) = \emptyset$.

Hence, the sequence $t\sigma \to^*_{U_{cs}(\mathcal{R})} v_{i+1}\sigma$ starts with a reduction step at position $p \in Pos_{\mathcal{F}}(t)$. We assume that the reduction sequence is non-empty, otherwise t and v_{i+1} would unify. Moreover, t is assumed to be linear. We show that there is a narrowing $\overline{t}$ of t obtained by narrowing t with mgu θ, such that $v_1 \to u_1, \dots v_i \to u_i, s\theta \to \overline{t}, v_{i+1} \to u_{i+1}, \dots$ is an infinite chain and each term in this chain can be instantiated such that it can be reduced to a $\mathcal{F}'$-term.

The reduction sequence $t\sigma \to^*_{U_{cs}(\mathcal{R})} v_{i+1}\sigma$ starts with a single reduction $t\sigma = t[l\rho]_p\sigma \to_{U_{cs}(\mathcal{R})} t[r\rho]_p\sigma$ using a rule $l \to r$. Since we consider l and t to be variable disjoint, we extend σ so that $x\sigma = x\rho$ for all $x \in Dom(\rho)$. Thus, σ unifies l and $t|_p$ and there is also an mgu θ for l and $t|_p$ ($\sigma = \tau \circ \theta$).

Then t narrows to $\overline{t} = t[r\theta]_p$ and since $s\theta \to \overline{t}$ is assumed to be variable disjoint from all other pairs in a chain, we can adapt σ to behave like τ on the variables of $s\theta$ and $\overline{t}$. Thus,

$$\begin{aligned} u_i\sigma \to^*_{U_{cs}(\mathcal{R})} s\sigma &= s\theta\tau = s\theta\sigma \\ \overline{t}\sigma = \overline{t}\tau = t[r\theta\tau]_p\theta\tau &= \sigma t[\sigma r]_p = \sigma t[r\rho]_p \to^*_{U_{cs}(\mathcal{R})} v_{i+1}\sigma \end{aligned}$$

and $v_1 \to u_1, \dots v_i \to u_i, s\theta \to \overline{t}, v_{i+1} \to u_{i+1}, \dots$ is an infinite chain. Moreover, an instance (obtained through σ) of each subterm of $\overline{t}$ is reducible to a $\mathcal{F}'$-term, because this was true for the chain we started with and all terms of the new chain occur already in the original one. Thus, we have shown that infinity of an SS-CS-DP-problem P implies infinity of the problem $Proc_{\text{RFN}}(P)$.

Completeness: Let $P = (\mathcal{P} \cup \{s \to t\}, \mathcal{R}, \mu, \mathcal{F}')$ be an SS-CS-DP-problem such that t is linear and does not unify with any left-hand side of a rule in $\mathcal{P}$, and let $(\mathcal{P} \cup \{s\theta_1 \to t_1, \dots s\theta_n \to t_n\}, \mathcal{R}, \mu, \mathcal{F}')$ be $Proc_{\text{RFN}}(P)$. We show that if $v_1 \to u_1, \dots, v_i \to u_i, s\theta_m \to t_m, v_{i+1} \to u_{i+1}, \dots$ is a $(\mathcal{P} \cup \{s\theta_1 \to t_1, \dots s\theta_n \to t_n\}, \mathcal{R}, \mu)$-chain for some $1 \leq m \leq n$, then $v_1 \to u_1, \dots, v_i \to u_i, s \to t, v_{i+1} \to u_{i+1}, \dots$ is a chain as well.

As $v_1 \to u_1, \dots, v_i \to u_i, s\theta_m \to t_m, v_{i+1} \to u_{i+1}, \dots$ is a chain, there is substitution a σ such that $u_j\sigma \to^*_{U_{cs}(\mathcal{R})} v_{j+1}\sigma$ for all $j > 0, j \neq i$, $u_i\sigma \to^*_{U_{cs}(\mathcal{R})} s\theta_m\sigma$ and $t_m\sigma \to^*_{U_{cs}(\mathcal{R})} v_{i+1}\sigma$.

As $s \to t$ does not share any variables with the rules $v_j \to u_j$ for all $j > 0$, we can define σ' to behave like $\theta\sigma$ on the variables of $s \to t$ and like σ on all other variables. Thus, we have

$$u_i\sigma' \to^*_{U_{cs}(\mathcal{R})} s\theta\sigma = s\sigma'$$

and because of $t\theta \to_{U_{cs}(\mathcal{R})} t_m$ (by the definition of context-sensitive narrowing) we get

$$t\sigma' = t\theta\sigma \to^*_{U_{cs}(\mathcal{R})} t_m\sigma' \to^*_{U_{cs}(\mathcal{R})} v_{i+1}\sigma'$$

Thus, $v_1 \to u_1, \dots, v_i \to u_i, s \to t, v_{i+1} \to u_{i+1}, \dots$ is a chain and we can construct a $(\mathcal{P} \cup \{s \to t\}, U_{cs}(\mathcal{R}, \mu)$-chain out of a $(\mathcal{P} \cup \{s\theta_1 \to t_1, \dots s\theta_n \to t_n\}, U_{cs}(\mathcal{R}, \mu)$-chain this way. $\square$

Note that the precondition of the narrowed dependency pair not containing variables that are forbidden in its left-hand side but allowed in its right-hand side is crucial as the following example illustrates.

Example 23. *Consider the DP problem $\mathcal{P} = (DP, \mathcal{R}, \mu, \mathcal{F})$ given by*

$$DP = \left\{ \begin{array}{rcl} t^{\#}(f(x)) & \rightarrow & t^{\#}(h(x)) \\ t^{\#}(b) & \rightarrow & t^{\#}(f(a)) \end{array} \right.$$

$$\mathcal{R} = \left\{ \begin{array}{rcl} a & \rightarrow & b \\ h(x) & \rightarrow & U(x,x) \\ U(x,x) & \rightarrow & x \end{array} \right.$$

$\mathcal{F} = \{a, b, f, h, t\}$ and $\mu(g) = \{1\}$ for all $g \in \{h, U, t, t^{\#}\}$, $\mu(g) = \emptyset$ for all $g \in \{f\}$. Note that $\mathcal{R}$ is the transformed version of the DCTRS $\{a \rightarrow b, h(x) \rightarrow x \Leftarrow x \rightarrow^* x\}$. $\mathcal{P}$ is infinite, because there exists an infinite DP chain:

$$t^{\#}(f(a)) \overset{\epsilon}{\rightarrow} t^{\#}(h(a)) \rightarrow_{\mu} t^{\#}(h(b)) \rightarrow_{\mu} t^{\#}(U(b,b)) \rightarrow_{\mu} t^{\#}(b) \overset{\epsilon}{\rightarrow} t^{\#}(f(a))$$

The right-hand side of the first pair is linear and it does not unify with a left-hand side of any other pair. However, there are forbidden variables in its left-hand side that occur replacing in the right-hand side. Narrowing the first pair and thus replacing it by $t^{\#}(f(x)) \rightarrow t^{\#}(U(x,x))$ would yield a finite DP problem. Thus the precondition $\overline{Var^{\mu}}(l) \cap Var^{\mu}(r) = \emptyset$ for the narrowed rule $l \rightarrow r \in DP$ cannot be dropped.

The second dependency pair processor makes use of backward narrowing.

Definition 13 (restricted backward narrowing). *Let $(DP, \mathcal{R}, \mu, \mathcal{F}')$ be an SS-CS-DP-problem with $\mathcal{R} = (\mathcal{F}, R)$. If $u_i \rightarrow v_i \in DP$, $\overline{Var^{\mu}}(v_i) \cap Var^{\mu}(u_i) = \emptyset$, u_i is not unifiable with any right-hand side of a rule in DP and u_i is linear, then $Proc_{\text{RBN}}$ yields a new SS-CS-DP-problem $(DP', \mathcal{R}, \mu, \mathcal{F}')$ where*

$$DP' = (DP - \{u_i \rightarrow v_i\}) \cup \{u_i^k \rightarrow v_i^k \theta_k \mid 1 \leq k \leq n\}$$

and $\{u_i^1, \ldots, u_i^n\}$ is the set of (one-step, context-sensitive) backward *narrowings of u_i with corresponding mgu's $\theta_1, \ldots, \theta_n$, such that all subterms of u_i^k are* reachable *from $\mathcal{F}'$-terms for all $1 \leq k \leq n$.*

Theorem 9. *The dependency pair processor $Proc_{\text{RBN}}$ is sound and complete for an SS-CS-DP-problem $(DP, U_{cs}(\mathcal{R}), \mu, \mathcal{F}')$ where $DP = (\mathcal{F}^{\sharp}, S^{\sharp})$ and $U_{cs}(\mathcal{R}) = (\mathcal{F}, R)$ provided that $(U_{cs}(\mathcal{R}), \mu')$ is obtained by the transformation of Definition 4 from some DCTRS $\mathcal{R}$ ($\mu'(f) = \mu(f)$ for all $f \in Dom(\mu')$) and $\mathcal{F}^{\sharp} \cap (\mathcal{F} \setminus \mathcal{F}') = \emptyset$ (i.e., $\mathcal{F}^{\sharp}$ does not contain any U-symbols).*

Proof. Analogous to the proof of Theorem 8. □

The narrowing processors use the notions *reducible to* respectively *reachable from* which are both undecidable in general. Thus, in order to apply these processors in practice, we need to use heuristics to approximate these notions. A very simple approach would be to discard only those narrowings that are U-terms and (forward resp. backward) narrowing normal forms. This heuristic is also used in the implementation of these processors in VMTL [82]. Note that when using approximations of the notions "reducible to" and "reachable from" the narrowing processors may no longer be complete (cf. Example 24), hence they cannot be used to prove non-termination on original terms in general.

Examples 24 and 25 below show that this simple approximation is already sufficient to prove termination on original terms where ordinary termination does not hold (Example 24), or to significantly reduce the number of narrowings that have to be considered (Example 25).

Apart from such simple approximations one could also think of more sophisticated ones. For instance in the "forward" approach non-reducibility to original terms could be detected by *root-stability*[6], which is still undecidable but for which non-trivial decidable approximations exist (e.g. strong root stability [49]).

Example 24. *Consider the transformed CSRS $\mathcal{R}$ of Example 21 and the SS-CS-DP-problem $\mathcal{P}_0 = (DP_0, \mathcal{R}, \mu, \mathcal{F}')$ where $DP_0 = DP(\mathcal{R}, \mu)$, μ has been extended to take dependency pair symbols into account and $\mathcal{F}'$ is $\mathcal{F}$ minus all U-symbols. $DP(\mathcal{R}, \mu) = \{f^\sharp(k(a), k(b), x) \to f^\sharp(x, x, x)\}$[7]. Applying $Proc_{\text{RBN}}$ to $\mathcal{P}_0$, we obtain a new problem $\mathcal{P}_1 = (DP_1, \mathcal{R}, \mu, \mathcal{F}')$ where*

$$\begin{aligned} DP_1 = \{f^\sharp(U_2(c(a), z), k(b), x) &\to f^\sharp(x, x, x), \\ f^\sharp(k(a), U_2(c(b), z), x) &\to f^\sharp(x, x, x)\}. \end{aligned}$$

$Proc_{\text{RBN}}$ can be applied again using either rule in DP_1 for narrowing. After iterated applications of $Proc_{\text{RBN}}$, all narrowings of left-hand sides of rules in DP_i contain the term $U_1(d, d)$ as their first or second argument. As this term is a backward narrowing normal form, $DP_{i+1} = \emptyset$ and we conclude termination on original terms according to Theorem 9.

Note that in this example it is critical to discard narrowings that contain the term $U(d, d)$, because this term is not reachable by an original term. If one used too rough approximations for reachability by original terms and considered terms containing $U(d, d)$ as valid terms appearing on DP chains, then indeed infinite DP-chains would exist. However, the conclusion that the system is non-μ-terminating on original terms would be incorrect, because when using approximations for the notion "reachable from" the backward narrowing processor is no longer complete.

[6]A term t is *root-stable* w.r.t. to a rewrite system $\mathcal{R}$ if there is no $\mathcal{R}$-reduction issuing from t that contains a root reduction step.

[7]Here, we restrict the set of dependency pairs to those that are possibly part of a cycle in the *dependency graph*. See [2] for a motivation and justification of this approach.

Example 25. *Consider the transformed CSRS $\mathcal{R}$ of Example 16. We use forward narrowing on the rule.*

$$A^\sharp \to h^\sharp(f(a), f(b))$$

Thus, the pair is replaced by two new rules

$$\begin{array}{lcl} A^\sharp & \to & h^\sharp(U(a,a), f(b)) \\ A^\sharp & \to & h^\sharp(f(a), U(b,b)) \end{array}$$

$Proc_{\text{RFN}}$ can be applied again to the resulting problem, such that the right-hand sides of the new rules are narrowed. Eventually, one of the arguments of $h^\sharp$ will narrow to instances of $U(d,x)$, $U(k,x)$, $U(l,x)$ or $U(m,x)$. As all instances of these terms are root-stable*, those narrowings can be disregarded according to Definition 12. Thus, in the row of SS-CS-dependency pair problems obtained by repeated application of $Proc_{\text{RFN}}$, the size of the TRSs (to be precise of the TRS in the first component of the tuples) will not grow as fast as it would, if no narrowings were discarded and smaller problems are obviously easier to handle (also with other dependency pair processors) than bigger ones. Indeed, termination of the CSRS of this example can be shown automatically with the described method (cf. Example 27 below).*

3.2.4.2 Instantiation Processors

In a sense, the transformation of Definition 4 distributes the evaluation of the conditions of one conditional rule among several unconditional rules. The results of these single evaluations are propagated through the variables from one unconditional rule to the next one. With our narrowing approach we try to approximate the results of single evaluations, but we still need a way to propagate these results in proofs of termination.

To this end we propose an *instantiation* processor, whose informal goal is to propagate the results of condition evaluations approximated through narrowing to subsequent conditions (i.e. subsequent rules in the transformed system).[8] The following lemma provides the theoretical basis for our instantiation processor.

Lemma 7. *Let $\mathcal{P} = (\mathcal{F}, R)$ and $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R')$ be TRSs with a combined replacement map μ. If $s\theta \overset{\epsilon}{\to}_{\mathcal{P},\mu} t\theta \overset{>\epsilon^*}{\to}_{\mathcal{R},\mu} s'\theta' \overset{\epsilon}{\to}_{\mathcal{P},\mu} t'\theta'$, $s'\sigma = t$ for some substitution σ, $\overline{Var^\mu}(t') \cap Var^\mu(s') = \emptyset$ and all variables of s' are contained only in constructor subterms (w.r.t. $\mathcal{R}$) (i.e. $s'|_p \in Var \Rightarrow \forall q < p\colon root(s'|_q) \in (\mathcal{F} \cup \mathcal{C}) \setminus \mathcal{D}$), then $s'\sigma\theta \overset{\epsilon}{\to}_{\mathcal{P},\mu} t'\sigma\overline{\theta} \to^*_{\mathcal{R},\mu} t'\theta'$ for some $\overline{\theta}$, such that $x\overline{\theta} = x\theta$ for all $x \in Var(t)$.*

Proof. Let $\{x_1, \ldots, x_n\}$ be the variables of t'. Assume x_i occurs in s' at position q. Then, we have that $x_i\sigma\theta \to^*_{\mathcal{R},\mu} x_i\theta'$, as $x_i\sigma\theta = t|_q\theta$ and $x_i\theta' = s'|_q\theta'$ and all positions

[8]Note that our instantiation processor is similar, but incomparable to the one in [33], as in [33] variables are only instantiated by constructor terms while according to Definition 14 in our approach also terms containing defined symbols can be substituted (cf. Example 26 below).

above q are constructors in t and s'. Thus, we set $y\overline{\theta} = y\theta$ for all $y \in \mathit{Var}(\mathit{Codom}(\sigma))$ and obtain $t'|_{q'}\sigma\overline{\theta} \rightarrow^*_{\mathcal{R},\mu} t'|_{q'}\theta'$ for any position q' with $t'|_{q'} = x_i$.

Note that if q is replacing in s', then so is q' in t'. Otherwise, $x_i\sigma\overline{\theta}S = x_i\theta'$.

Hence, $s'\sigma\overline{\theta} \overset{\epsilon}{\rightarrow}_{\mathcal{P},\mu} t'\sigma\overline{\theta}$ and we have that $x\sigma\overline{\theta} \rightarrow^*_{\mathcal{R},\mu} x\theta'$ for all $x \in t'$ and thus $t'\sigma\overline{\theta} \rightarrow^*_{\mathcal{R},\mu} t'\theta'$. □

Definition 14 (backward instantiation processor)**.** *Let $(DP = \{s \rightarrow t\} \cup DP', \mathcal{R}, \mu, \mathcal{F}')$ be an SS-CS-DP-problem with $\mathcal{R} = (\mathcal{F}, R)$, such that all variables of s are contained only in constructor subterms of s (w.r.t. $\mathcal{R}$) and $\overline{\mathit{Var}^\mu}(t) \cap \mathit{Var}^\mu(s) = \emptyset$. The set $\mathit{Pred}_{s\rightarrow t} = \{l \rightarrow r \in DP \mid \exists\theta.\mathit{cap}(\mathit{ren}(r))\theta = \mathit{cap}(\mathit{ren}(s))\theta\}$ defines all potential predecessors of the pair $s \rightarrow t$ on $(DP, \mathcal{R}, \mu)$-chains.*[9] *If, for all $l \rightarrow r \in \mathit{Pred}_{s\rightarrow t}$, $r = s\sigma$ for some σ, then the processor $\mathit{Proc}_{\text{BI}}$ yields $(DP' \cup \{s\sigma \rightarrow t\sigma \mid l \rightarrow r \in \mathit{Pred}_{s\rightarrow t} \wedge r = s\sigma\}, \mathcal{R}, \mu, \mathcal{F}')$.*

Theorem 10. *The processor $\mathit{Proc}_{\text{BI}}$ is sound and complete.*

Proof. Soundness: Assume there is an infinite dependency pair chain w.r.t. to a DP problem $\mathcal{P} = (DP, \mathcal{R}, \mu, \mathcal{F})$. We show that there also exists an infinite chain w.r.t. to the problem $\mathit{Proc}_{\text{BI}}(\mathcal{P}) = \mathcal{P}'$.

Consider an arbitrary fragment of the initial infinite chain:

$$\ldots t_i\theta \rightarrow^*_{\mathcal{R},\mu} s_{i+1}\theta' \overset{\epsilon}{\rightarrow}_{DP} t_{i+1}\theta' \ldots .$$

Then, we can construct an analogous chain fragment in $\mathit{Proc}_{\text{BI}}(\mathcal{P})$, as either $s_{i+1} \rightarrow t_{i+1}$ is contained in the dependency pairs of the derived problem $\mathcal{P}'$, or $t_i = s_{i+1}\sigma$ and thus there is a dependency pair $s_{i+1}\sigma \rightarrow t_{i+1}\sigma$ in $\mathcal{P}'$. In the latter case the new chain fragment is

$$\ldots t_i\theta = s_{i+1}\sigma\overline{\theta} \overset{\epsilon}{\rightarrow}_{\mathcal{P}'} t_{i+1}\sigma\overline{\theta} \rightarrow^*_{\mathcal{R},\mu} t_{i+1}\theta'$$

for some $\overline{\theta}$ (according to Lemma 7).

Completeness: Consider an infinite chain w.r.t. $\mathcal{P}'$. $\ldots s_i\sigma\theta \overset{\epsilon}{\rightarrow} t_i\sigma\theta \ldots$. As we assume that all dependency pairs in chains are variable disjoint we can adapt θ to behave like $\sigma\theta$ and thus obtain an infinite DP chain w.r.t. to the original problem $\mathcal{P}$. □

Example 26. *Consider an SS-CS-DP-problem $P = (DP, \mathcal{R}, \mu, \mathcal{F}')$ where*

$$DP = \left\{ \begin{array}{rcl} d^\# & \rightarrow & U_1^\#(c) \\ U_1^\#(x) & \rightarrow & c^\# \end{array} \right.$$

$$\mathcal{R} = \left\{ \begin{array}{rcl} d & \rightarrow & U_1(c) \\ U_1(x) & \rightarrow & c \\ c & \rightarrow & b \end{array} \right.$$

[9] To be precise this definition of $\mathit{Pred}_{s\rightarrow t}$ identifies a superset of potential predecessor pairs of $s \rightarrow t$ in DP chains. The exact set is in general undecidable, however one could use other/better approximations here as well.

$\mu(U_1^\#) = \mu(U_1) = \{1\}$ and $\mathcal{F}' = \{c, d\}$. The problem originates from the dependency pair analysis of the DCTRS $\mathcal{R}$:

$$\begin{array}{rcl} d & \to & x \Leftarrow c \to^* x \\ c & \to & b \end{array}$$

The backward instantiation processor can be applied to P. The dependency pair $s \to t$ is $U_1^\#(x) \to x$ and its only potential ancestor is $d^\# \to U_1^\#(c)$. Since all functions in s above the variable x are constructors (i.e. x is contained in a constructor context in s) and the variable of t is replacing (i.e. $\overline{Var}^\mu(t) = \emptyset$), the additional preconditions for the application of the processor are satisfied. Thus, according to Definition 14 the result of the application of the processor is a new dependency pair problem $(DP', \mathcal{R}, \mu, \mathcal{F}')$ where

$$DP' = \left\{ \begin{array}{rcl} d^\# & \to & U_1^\#(c) \\ U_1^\#(c) & \to & c^\# \end{array} \right.$$

Note that finiteness of this resulting SS-CS-DP-problem is obvious and can easily be shown be repeated application of the forward narrowing processor of Definition 12.

Example 27. *Inside the dependency pair framework termination on original terms of $U_{cs}(\mathcal{R})$ and thus operational termination of $\mathcal{R}$ for the DCTRS $\mathcal{R}$ from Example 16 can be proved by repeated application of forward narrowing and backward instantiation. Our experiments showed that μ-termination of $U_{cs}(\mathcal{R})$ is hard to prove using other, standard techniques for termination analysis, thus the introduced dependency pair processors seem crucial for this particular example.*

Analogously to the backward instantiation processor we can also define a processor for forward instantiation.

Definition 15 (forward instantiation processor)**.** *Let $(DP = \{s \to t\} \cup DP', \mathcal{R}, \mu, \mathcal{F}')$ be an SS-CS-DP-problem with $\mathcal{R} = (\mathcal{F}, R)$, such that all variables of t are contained only in constructor subterms of t (w.r.t. $\mathcal{R}$) and $Var^\mu(t) \cap \overline{Var}^\mu(s) = \emptyset$. The set $Succ_{s\to t} = \{l \to r \in DP \mid \exists\theta.cap(ren(t))\theta = cap(ren(l))\theta\}$ defines all potential successors of the pair $s \to t$ on $(DP, \mathcal{R}, \mu)$-chains.*[10] *If, for all $l \to r \in Succ_{s\to t}$, $l = t\sigma$ for some σ, then the processor $Proc_{\text{FI}}$ yields $(DP' \cup \{s\sigma \to t\sigma \mid l \to r \in Succ_{s\to t} \wedge l = t\sigma\}, \mathcal{R}, \mu, \mathcal{F}')$.*

In order to prove soundness and completeness we proceed as for the backward instantiation processor and show the following lemma that is dual to Lemma 7.

[10]To be precise this definition of $Succ_{s\to t}$ identifies a superset of potential successor pairs of $s \to t$ in DP chains. The exact set is in general undecidable, however one could use other/better approximations here as well.

Lemma 8. *Let $\mathcal{P} = (\mathcal{F}, R)$ and $\mathcal{R} = (\mathcal{D} \uplus \mathcal{C}, R')$ be TRSs with a combined replacement map μ. If $s\theta \overset{\epsilon}{\to}_{\mathcal{P},\mu} t\theta \overset{>\epsilon^*}{\to}_{\mathcal{R},\mu} s'\theta' \overset{\epsilon}{\to}_{\mathcal{P},\mu} t'\theta'$, $t\sigma = s'$ for some substitution σ, $\overline{Var}^{\mu}(s) \cap Var^{\mu}(t) = \emptyset$ and all variables of t are contained only in constructor subterms (w.r.t. $\mathcal{R}$) (i.e. $t|_p \in Var \Rightarrow \forall q < p$: $root(t|_q) \in (\mathcal{F} \cup \mathcal{C}) \setminus \mathcal{D}$), then $s\theta \to^*_{\mathcal{R},\mu} s\sigma\overline{\theta}$ for some $\overline{\theta}$, such that $x\overline{\theta} = x\theta'$ for all $x \in Var(s')$.*

Proof. Let $\{x_1, \dots, x_n\}$ be the variables of s. We distinguish two cases for each variable x_i. First, assume x_i occurs in t at position q. Then, we have that $x_i\theta \to^*_{\mathcal{R},\mu} x_i\sigma\theta'$, as $x_i\theta = t|_q\theta$, $x_i\sigma\theta' = s'|_q\theta'$ and all positions above q are constructors in t and s'. Thus, we set $y\overline{\theta} = y\theta'$ for all $y \in Var(Codomain(\sigma))$ and obtain $s|_{q'}\theta \to^*_{\mathcal{R},\mu} s|_{q'}\sigma\overline{\theta}$ for any position q' with $s|_{q'} = x_i$. Note that if q is replacing in t, then so is q' in s. Otherwise, $x_i\theta = x_i\sigma\overline{\theta}$.

Secondly, if x_i does not occur in t, then it does neither occur in $Dom(\sigma)$ nor in $Var(Codomain(\sigma))$. Thus, we set $x_i\overline{\theta} = x_i\theta$ and obtain $s|_p\theta = s|_p\sigma\overline{\theta}$ for any position p with $s|_p = x_i$. □

Theorem 11. *The processor $Proc_{\text{FI}}$ is sound and complete.*

Proof. Soundness: Assume there is an infinite dependency pair chain w.r.t. to a DP problem $\mathcal{P} = (DP, \mathcal{R}, \mu, \mathcal{F})$. We show that there also exists an infinite chain w.r.t. to the problem $Proc_{\text{FI}}(\mathcal{P}) = \mathcal{P}'$.

Consider an arbitrary fragment of the initial infinite chain:

$$\dots s_i\theta \overset{\epsilon}{\to}_{DP} t_i\theta \to^*_{\mathcal{R},\mu} s_{i+1}\theta' \dots .$$

Then, we can construct an analogous chain fragment in $Proc_{\text{FI}}(\mathcal{P})$, as either $s_i \to t_i$ is contained in the dependency pairs of the derived problem $\mathcal{P}'$, or $s_{i+1} = t_i\sigma$ and thus there is a dependency pair $s_i\sigma \to t_i\sigma$ in $\mathcal{P}'$. In the latter case the new chain fragment is

$$\dots s_i\theta \to^*_{\mathcal{R},\mu} s_i\sigma\overline{\theta} \overset{\epsilon}{\to}_{\mathcal{P}'} t_i\sigma\overline{\theta} = s_{i+1}\theta'$$

(according to Lemma 8).

Completeness: Consider an infinite chain w.r.t. $\mathcal{P}'$. $\dots s_i\sigma\theta \overset{\epsilon}{\to} t_i\sigma\theta \dots$. As we assume that all dependency pairs in chains are variable disjoint, we can adapt θ to behave like $\sigma\theta$ and thus obtain an infinite DP chain w.r.t. to the original problem $\mathcal{P}$. □

Note that the narrowing and instantiation approach is just one out of many methods to analyze dependency pair problems for their finiteness in the setting of ordinary termination analysis. However, regarding the structure of the systems that we analyze and using the fact that they were obtained from DCTRSs, narrowing and instantiation seem to be an adequate tool in our special setting, because they are in some cases able to identify those instances of left-hand sides of rules for which the conditions of the corresponding DCTRS are satisfiable.

Taking into account that finding such instances or identifying instances for which the conditions are not satisfiable is potentially crucial for proving or disproving termination of (transformed) systems, narrowing and instantiation are important tools for this task. Moreover, our narrowing dependency pair processors allow us to reduce the number of narrowings generated and thus make the narrowing approach more efficient in practice.

In the experiments we performed to evaluate our approach, the combination of narrowing and instantiation was only part of the strategy for finding proofs in the dependency pair framework. More precisely, we applied the narrowing processors (backward and forward in parallel; cf. [82, Section 3.1]) until they were no longer applicable and used the instantiation processors afterwards. See Section 3.3 for details on other DP processors available in our tool VMTL.

Example 28. *In Example 24, after several narrowing steps the first TRS of the SS-CS-DP-problem is empty, thus the conditions of the conditional rule are unsatisfiable. Note that this DCTRS $\mathcal{R}$ is operationally terminating while $U_{cs}(\mathcal{R})$ is not μ-terminating. Hence, operational termination cannot be verified with standard ordering-based methods. Thus, again the presented narrowing processor is crucial for a successful automatic proof of operational termination.*

3.3 Implementation and Evaluation

In order to evaluate the practical use of the context-sensitive unraveling as well as our approach to prove termination on restricted sets of terms, we implemented both the transformation and our proposed dependency pair processors in the tool VMTL (cf. [82] and also Chapter 5 of this thesis). In addition to the DP processors introduced in Section 3.2.4 VMTL contains implementations of various standard (mostly ordering based) DP processors. In addition, a simple check for infinity of DP problems is included that can be viewed as a DP processor returning "`no`", hence enabling VMTL to prove non-termination. Note that, as the narrowing processors (using approximations for deciding reducibility to resp. reachability from) are not complete, infinity of a DP problem does not imply non-termination of the original rewrite system on original terms after they have been used during the proof search. The results and details of our tests can be found at the tool's homepage.[1] Out of 27 tested systems our implementation was able to prove operational termination of 17. Note that for only one DCTRS in this collection the transformed system is not μ-terminating on all, but only on original terms (i.e. Example 21). However, we refrained from providing more examples of this kind, since we conjecture that they would all have a structure similar to the DCTRS in Example 21. This conjecture is supported by the fact that for such TRSs $\mathcal{R}$ (i.e., where $\mathcal{R}$ is operationally terminating while $U_{cs}(\mathcal{R})$ is non-μ-terminating), $\mathcal{R} \cup C_{\mathcal{E}}$ is not operationally terminating (cf. Theorem 6 and Corollary 4). Moreover, DCTRSs with this property are rather pathological and do not arise naturally as program specifications. We showed, however, that our approach is useful also for proving termination of DCTRSs not belonging to this class. This is supported by our experiments where operational termination of several DCTRSs $\mathcal{R}$ could be shown whereas they could not be handled by traditional methods despite $\mathcal{R} \cup C_{\mathcal{E}}$ being operationally terminating as well.

The examples used in the experiments were taken from the termination problem database (TPDB, cf. [1]) and from standard literature on conditional term rewriting (e.g. [73] and [64]).

In our experiments other termination tools supporting conditional rewrite systems scored worse on this set of examples. For instance, the tool AProVE [31] was able to prove operational termination of 15 examples through the web-interface. However, the batch version (i.e. AProVE 1.2[11]) could only prove operational termination of 12 examples. This illustrates that termination of CSRSs obtained by our transformation may be hard to verify, and sophisticated and complicated proof methods (as implemented only in the most recent version of AProVE) may be needed.

Overall, VMTL was able to prove operational termination of 6 DCTRSs for which AProVE failed. On the other hand, operational termination of 4 other DCTRSs could only be successfully proved by AProVE. In the 6 examples where VMTL was successful while AProVE was not, the narrowing and instantiation processors of Section 3.2.4 played a crucial rule.

On the negative side, repeated application especially of narrowing processors can be expensive with respect to execution time (and space). Yet, we did not restrict the application of the narrowing and instantiation processors by imposing complex applicability conditions as, for instance, described in [33, Section 5.2] using the concept of *safe transformations*. The reason is that for the particular class of rewrite systems obtained by transformations from conditional systems it might be necessary to spend more time on narrowing and instantiation techniques than on the search for applicable orderings. Still such applicability conditions tailored to systems obtained by the transformation of Definition 4 would be an interesting direction for future work.

Note also that inside the dependency pair framework DP processors may be applied to DP-problems in an arbitrary order. Choosing and fixing such an order can significantly influence the power and efficiency of a termination tool. In our experiments, the narrowing and instantiation approach was only tried after other ordering-based methods to prove finiteness of DP-problems, which are more efficient, failed. This strategy turned out to be the most efficient and powerful one.

3.4 Related Work

The idea of using context-sensitivity to improve the unraveling transformation of [64, 63, 72, 73] is not new. In [20, 70, 18] the same idea is used in conjunction with another optimization. The second optimization is to store the bindings of only those variables in the arguments of a U_j^α symbol that occur in a subsequent condition or

[11]Newer batch versions of AProVE failed to prove termination of any DCTRSs with extra variables in our experiments.

in the right-hand side of the rule α.

For clarity we provide a formal definition of this optimization.

Definition 16 (optimized transformation according to [20, 70, 18])**.** *Let $\mathcal{R}$ be a DCTRS ($\mathcal{R} = (\mathcal{F}, R)$). For every rule $\alpha\colon l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_n \to^* t_n$ we use n new function symbols U_i^α ($i \in \{1, \ldots, n\}$). Then α is transformed into a set of unconditional rules in the following way:*

$$\begin{array}{rcl} l & \to & U_1^\alpha(s_1, \vec{x}_1) \\ U_1^\alpha(t_1, \vec{x}_1) & \to & U_2^\alpha(s_2, \vec{x}_2) \\ & \vdots & \\ U_n^\alpha(t_n, \vec{x}_n) & \to & r \end{array}$$

Here the sequences of variables $\vec{x}_i$ are given by (an arbitrary but fixed sequentialization of the set of variables)

$$(Var(l) \cup Var(t_1) \ldots Var(t_{i-1})) \cap$$
$$(Var(t_i) \cup Var(s_{i+1}) \cup Var(t_{i+1}) \ldots Var(s_n) \cup Var(t_n) \cup Var(r)).$$

The transformed system $U_{opt}(\mathcal{R}) = (U(\mathcal{F}), U_{opt}(R))$ is obtained by transforming each rule of $\mathcal{R}$ where $U(\mathcal{F})$ is $\mathcal{F}$ extended by all new function symbols. We use a replacement map μ_{opt} given by $\mu_{opt}(U) = \{1\}$ for every auxiliary symbol U (i.e. $U \in U(\mathcal{F}) \setminus \mathcal{F}$) and $\mu_{opt}(f) = \{1, \ldots, ar(f)\}$ for every $f \in \mathcal{F}$.

Indeed, according to [18] it holds that whenever $U_{opt}(\mathcal{R})$ is μ_{opt}-terminating, $\mathcal{R}$ is operationally terminating.[12] [13] Since the transformation of Definition 16 produces smaller transformed systems than the one from Definition 4, it might be advantageous to use it in termination analysis. However, there is a price to pay for this optimization. That is, one loses the property of simulation-soundness (cf. Theorem 2).

Example 29. *Consider a DCTRS $\mathcal{R}$ given by*

$$\begin{array}{rcl} f(x) & \to & c \Leftarrow a \to^* b \\ g(x, x) & \to & g(f(a), f(b)) \end{array}$$

[12]The transformation we presented in Definition 16 is actually a special case of the transformation introduced in [18]. There, the authors work in a more general setting where $\mathcal{R}$ itself may be context-sensitive and rewriting modulo an equational theory is used.

[13]Note, however, that in [18, p. 78] the authors introduce both $U_{cs}(\mathcal{R})$ and $U_{opt}(\mathcal{R})$, but do not clearly distinguish between them subsequently. This appears to be justified in the context of [18, Theorem 2 and Lemma 3] (because the proofs of these latter results work for both versions of the transformation), but not in general, since the two transformations have different properties, cf. Examples 29, 30. In particular, μ_{opt}-termination of $U_{opt}(\mathcal{R})$ implies $\mu_{U_{cs}(\mathcal{R})}$-termination of $U_{cs}(\mathcal{R})$, but not vice versa.

The transformed system $U_{cs}(\mathcal{R})$ consists of the following rules

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

$U_{cs}(\mathcal{R})$ is μ-terminating and thus $\mathcal{R}$ is operationally terminating. However, $U_{opt}(\mathcal{R})$ given by

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

is easily seen to be non-(μ_{opt}-)terminating (even on original terms) due to the cyclic reduction sequence

$$g(f(a),f(b)) \to^{+}_{U_{opt}(\mathcal{R})} g(U(a),U(a)) \to_{U_{opt}(\mathcal{R})} g(f(a),f(b))$$

Hence, Theorems 2 and 4 and Corollary 4 do not hold for this optimized transformation.[14]

Note that the DCTRS in Example 29 is not left-linear. However, this property is not crucial as the following left-linear example shows (also the left-hand sides of conditions are linear).

Example 30. *Consider the DCTRS $\mathcal{R}$ given by*

$$\begin{array}{rcl} f(x) & \to & y \Leftarrow a \to^* h(y) \\ g(x,b) & \to & g(f(c),x) \Leftarrow f(b) \to^* x, x \to^* c \\ a & \to & h(b) \\ a & \to & h(c) \end{array}$$

$U_{opt}(\mathcal{R})$ is given by

$$\begin{array}{rcl} f(x) & \to & U_f(a) \\ U_f(h(y)) & \to & y \\ g(x,b) & \to & U_g^1(f(b),x) \\ U_g^1(x,x) & \to & U_g^2(x,x) \\ U_g^2(c,x) & \to & g(f(c),x) \\ a & \to & h(b) \\ a & \to & h(c) \end{array}$$

[14]Technically, this is reflected in the fact that the the back-translation function `tb` (which is crucial for the proofs of these results) according to Definition 3 would not be well-defined for $U_{opt}(\mathcal{R})$ instead of $U_{cs}(\mathcal{R})$. The reason is that the substitution σ involved in Definition 3 may become non-well-defined due to the existence of erasing rules in $U_{opt}(\mathcal{R})$ that forget certain variable bindings (cf. the first rule of $U_{opt}(\mathcal{R})$ in Examples 29 and 30, respectively). For a more general and thorough discussion of (desirable) properties of back-translation functions in the setting of transforming CTRSs into TRSs we refer to [34].

Then $\mathcal{R}$ is operationally terminating (however, this has been proved by hand – via analyzing the shape of potentially existing minimal counterexamples – as automated termination tools currently fail to prove μ-termination of $U_{cs}(\mathcal{R})$), but $U_{opt}(\mathcal{R})$ is again non-terminating due to the following cyclic reduction sequence.

$$\begin{aligned} g(f(c),b) &\to g(U_f(a),b) \to U_g^1(f(b),U_f(a)) \to U_g^1(U_f(a),U_f(a)) \\ &\to U_g^2(U_f(a),U_f(a)) \to U_g^2(U_f(h(c)),U_f(a)) \to U_g^2(c,U_f(a)) \\ &\to g(f(c),U_f(a)) \to g(f(c),U_f(h(b))) \to g(f(c),b) \end{aligned}$$

In [70] it is shown that left-linearity (and right linearity in combination with non-erasingness) of the transformed system $U_{opt}(\mathcal{R})$ is sufficient to guarantee simulation-soundness (even if context-sensitivity is dropped).

However, despite being an interesting question we refrain from giving a more precise assessment of conditions under which the optimized transformation is simulation sound. Yet, solving this problem could also be useful in practice, because automated termination provers could base the decision on which transformation to use on this knowledge.

In [70] simulation-soundness is obtained by restricting $U_{opt}(\mathcal{R})$-evaluations.[15] The idea is to contract only redexes not containing auxiliary U-symbols. That is to say that a term s rewrites to a term t under this restriction iff $s \xrightarrow{p}_{U_{opt}(\mathcal{R})} t$ and $s|_p$ is a term over the original signature of $\mathcal{R}$. In order to prove operational termination of a conditional system $\mathcal{R}$ it would be sufficient to prove termination of $U_{opt}(\mathcal{R})$ under this restriction. However, while this might be feasible, given the recent advances in proving termination under strategies (cf. e.g. [25]), no concrete methods for this particular task exist to the author's knowledge.

[15]Note that our notion of simulation-soundness is called simulation-completeness there.

Chapter 4

Generalizing Context-Sensitivity

4.1 Introduction and Related Work

The idea of using some form of evaluation strategies and replacement restrictions in order to obtain improved termination properties as compared to naive evaluation dates back to [28, 44]. The concept of lazy evaluation proposed there is an integral feature of many modern functional and functional logic programming languages (e.g. Haskell [47]). The basic idea of lazy evaluation is to evaluate arguments of functions only if they are needed to compute the function and to delay their evaluation otherwise.

Example 31. *Consider the following TRS*

$$\begin{aligned} if(true, x, y) &\rightarrow x \\ if(false, x, y) &\rightarrow y \end{aligned}$$

Here, when evaluating a term $if(c, s, t)$, where s and t are potentially non-normalizing terms, it is not desirable to compute the (possibly non-existing) results of s and t before knowing which result is actually needed (depending on c). Instead it is preferable to delay the evaluation of s and t until the value of c is known and thus one of the rewrite rules is applicable.

A crucial question that immediately arises when considering Example 31 is how to decide which arguments of a given function are to be evaluated eagerly, i.e. before the function itself is computed and which are to be evaluated in a lazy fashion i.e. after the function is computed. Usually, this problem is addressed by some kind of strictness analysis. This is to say that the arguments of each function are partitioned into strict and non-strict ones, where the values of the strict ones are crucial to compute the function (such as the first argument of the if function in Example 31) while non-strict arguments are (potentially) not crucial (such as the second and third arguments of the if function in Example 31).

Unfortunately, strictness is not decidable in general and hence approximations are needed to identify a superset of strict arguments for a given function. Canonical context-sensitive rewriting can be viewed as such an approximation for left-linear rewrite systems. However, it turned out that for some classes of TRS the formalism of context-sensitive rewriting is not fine-grained enough to obtain replacement restrictions with the desired properties.

Example 32 (e.g. [58]). *Consider the following TRS $\mathcal{R}$.*

$$\begin{array}{rcl} from(x) & \to & x : from(s(x)) \\ 2nd(x : (y : zs)) & \to & y \end{array}$$

Here, it is crucial to allow reduction in the single argument of $2nd$, *e.g. to compute the normal form of the term* $2nd(from(0))$ *(which is* $s(0)$*). On the other hand one would like to impose some replacement restrictions in order to obtain a terminating rewrite relation.*

On-demand Rewriting

Example 32 was used as the main motivating example of *on-demand rewriting* in [58, 56]. On-demand rewriting is an extension of the formalism of context-sensitive rewriting where an additional replacement map is used (so in total there are two replacement maps). Hence, the positions in a term are partitioned into three classes, the replacing ones, the forbidden ones and a third set of positions where evaluations may take place only if there is a demand. Roughly, a rewrite step is demanded if it contributes to a more outer function evaluation by creating a (part of a) redex.

While on-demand rewriting allows to impose more fine-grained restrictions of rewriting than context-sensitive rewriting and is a conservative extension of context-sensitive rewriting, its formal definition is rather complex and the analysis of TRSs restricted by on-demand rewriting is hard. In [58, Section 4] a practical approach for the automated termination analysis of on-demand TRSs based on transformations into unrestricted TRSs was proposed. One of the proposed transformations was even complete, meaning that termination of an on-demand TRS completely coincides with termination of the transformed (unrestricted) TRS. However, the transformations utilized there are similar in structure to the ones from [30], which are used for transforming context-sensitive TRSs into ordinary TRSs while preserving non-termination.

While these transformations enjoy nice theoretical properties, in practical experiments it turned out that they are of limited use in the automated termination analysis, because a considerable amount of syntax and structural complexity is introduced by them (this was observed for the first time in [13] and was empirically supported by the results of a yearly competition of automated termination tools analyzing a set of context-sensitive rewrite systems for termination, where, starting around 2007, direct methods began to outperform approaches based on transformations of CSRSs into TRSs).

Lazy Rewriting

Another approach to laziness in term rewriting is *lazy rewriting* of [27]. There, all positions in a term are labelled with either a lazy (l) or eager an (e) flag. Rewrite steps may only be performed at positions labelled with e. Initially, the labelling is established by a replacement map. However, the labels of l-labelled positions may change to e if a rewrite step at such a position might contribute to the creation of a redex at a more outer e-labelled position.

While the idea of lazy rewriting (i.e. to allow rewrite steps at otherwise forbidden positions if they can contribute to the creation of a more outer redex) is similar to the one of on-demand rewriting they are "in general not (easily) comparable" ([56, Section 5.3]). However, like on-demand rewriting, lazy rewriting is general enough to enforce termination in the critical Examples 7 and 32, while normal forms can still be computed.

The drawback of using lazy rewriting is the subtlety involved in the dynamic character of the labelings that also leads to quite complex and practically inefficient methods for the automated termination analysis of TRSs restricted by lazy rewriting (see e.g. [79, Example 2.8, 3.3 and Definition 3.6]).

Other approaches to restrictions of rewriting, that also fix the order of the argument evaluation of functions (instead of just partitioning the arguments into eager and lazy) such as rewriting with on-demand strategy annotations of [7], tend to suffer from the same problems as on-demand rewriting and lazy rewriting of having complex formal definitions and being hard to analyze. We refer to [77] for a more in-depth analysis and comparison of the mentioned approaches to lazy evaluation in term rewriting also with context-sensitive rewriting.

In the following we introduce the novel approach of forbidden pattern restrictions to rewriting that we hope can avoid some of the drawbacks of other notions of restricted rewriting while being equally (or even more) powerful.

4.2 Forbidden Patterns

4.2.1 Basic Definitions

The basic idea of forbidden patterns is to explicitly forbid reduction steps when the corresponding redex appears in a certain context and has a certain shape expressed by a forbidden pattern. Hence, technically rewriting with forbidden patterns only relies on matching and comparison of positions which makes it definitorily simpler and more intuitive than approaches like on-demand rewriting and lazy rewriting.

A unique feature of rewriting with forbidden patterns as compared to other approaches is that forbidden patterns explicitly specify the forbidden part of terms resp. the rewrite relation while other approaches typically make the allowed part explicit (e.g. through a replacement map). This is an important design decision which is based on the assessment that in many cases comparatively few surgical restrictions to a non-terminating rewrite relation are sufficient to make it terminating. Hence, specifying these restrictions explicitly seems to be more economic than implicit definitions through the complement of allowed rewrite steps.

The forbidden patterns that we use to specify restrictions are triples consisting of a term, a position and a flag.

Definition 17 (forbidden pattern)**.** *A* forbidden pattern *(w.r.t. to a signature $\mathcal{F}$) is a triple $\langle t, p, \lambda \rangle$, where $t \in \mathcal{T}(\mathcal{F}, V)$ is a term, $p \in Pos(t)$ and $\lambda \in \{h, b, a\}$.*

The intended meaning of the last component λ is to indicate whether the pattern forbids reductions

- exactly at position p, but not outside (i.e., strictly above or parallel to p) or strictly below – (h for here), or
- strictly below p, but not at or outside p – (b for below), or
- strictly above position p, but not at, below or parallel to p – (a for above).

Hence, the forbidden entities of rewriting with forbidden patterns can be either positions, subterms or contexts at resp. in which no reductions may occur. While this offers a great deal of flexibility in the design of actual forbidden pattern restrictions, the analysis of the resulting rewrite relation (e.g. regarding termination) tends to be hard in the simultaneous presence of all flags in a set of forbidden patterns. Hence, in the sequel, when providing effective methods for the analysis of rewrite systems restricted by forbidden patterns, we usually consider subclasses of forbidden patterns where the third component of forbidden patterns is restricted (e.g. to h as in Section 4.4.1 below).

Moreover, abusing notation we sometimes say a forbidden pattern is linear, unifies with some term etc. when we actually mean that the term in the first component of a forbidden pattern has this property.

We denote a *finite* set of forbidden patterns for a signature $\mathcal{F}$ by $\Pi_{\mathcal{F}}$ or just Π if $\mathcal{F}$ is clear from the context or irrelevant. For brevity, patterns of the shape $\langle \_, \_, h/b/a \rangle$ are also called $h/b/a$-patterns, or *here/below/above*-patterns.[1]

Note that if for a given term t we want to specify more than just one restriction by a forbidden pattern, this can easily be achieved by having several triples of the shape $\langle t, \_, \_ \rangle$.

As already mentioned, in contrast to other approaches to restricted rewriting, forbidden patterns are supposed to explicitly define the *forbidden* part of a rewrite relation, thus implicitly yielding allowed reduction steps as those that are not forbidden.

Definition 18 (forbidden pattern reduction relation)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS with forbidden patterns $\Pi_{\mathcal{F}}$. The* forbidden pattern reduction relation *$\to_{\mathcal{R},\Pi_{\mathcal{F}}}$, or $\to_{\Pi}$ for short is given by $s \to_{\mathcal{R},\Pi_{\mathcal{F}}} t$ if $s \xrightarrow{p}_{\mathcal{R}} t$ for some $p \in Pos_{\mathcal{F}}(s)$ such that there is no pattern $\langle u, q, \lambda \rangle \in \Pi_{\mathcal{F}}$, no context C and no position q' with*

[1]Here and subsequently we use a wildcard notation for forbidden patterns. For instance, $\langle \_, \_, i \rangle$ stands for $\langle t, p, i \rangle$ where t is some term and p some position in t of no further relevance.

- *$s = C[u\sigma]_{q'}$ and $p = q'.q$, if $\lambda = h$,*
- *$s = C[u\sigma]_{q'}$ and $p > q'.q$, if $\lambda = b$, and*
- *$s = C[u\sigma]_{q'}$ and $p < q'.q$, if $\lambda = a$.*

Note that for a finite rewrite system $\mathcal{R}$ (with finite signature $\mathcal{F}$) and a finite set of forbidden patterns $\Pi_\mathcal{F}$ it is decidable whether $s \to_{\mathcal{R},\Pi_\mathcal{F}} t$ for terms s and t. We write $(\mathcal{R}, \Pi)$ for rewrite systems with associated forbidden patterns. Such a rewrite system $(\mathcal{R}, \Pi)$ is said to be Π-terminating (or just terminating if no confusion arises) if $\to_{\mathcal{R},\Pi}$ is well-founded. We also speak of Π-normal forms instead of $\to_{\mathcal{R},\Pi}$-normal forms. We denote by $Pos^\Pi(t)$ the set of those positions of t that are allowed and by $\overline{Pos}^\Pi(t)$ those that are forbidden according to a set Π of forbidden patterns.

Special degenerate cases of $(\mathcal{R}, \Pi)$ include e.g. $\Pi = \emptyset$ where $\to_{\mathcal{R},\Pi} = \to_\mathcal{R}$, and $\Pi = \{\langle l, \epsilon, h\rangle \mid l \to r \in R\}$ where $\to_{\mathcal{R},\Pi} = \emptyset$.

In the sequel we use the notions of *allowed* and *forbidden* (by Π) redexes. A redex $s|_p$ of a term s is allowed if $s \xrightarrow{p}_\Pi t$ for some term t, and forbidden otherwise.

Example 33. *Consider the TRS of Example 32. If we use a set of forbidden patterns Π given by $\{\langle x : (y : from(z)), 2.2, h\rangle\}$ the relation $\to_\Pi$ is terminating (and termination can be shown automatically, cf. Example 44 below). Moreover, $\to_\Pi$ is powerful enough to compute original head-normal forms if they exist (cf. Example 66 below).*

Example 34. *Consider the non-terminating many-sorted TRS $\mathcal{R}$ given by*

$$\begin{aligned} take(0, y : ys) &\to y & app(nil, ys) &\to ys \\ take(s(x), y : ys) &\to take(x, ys) & app(x : xs, ys) &\to x : app(xs, ys) \\ take(x, nil) &\to 0 & from(x) &\to from(s(x)) \end{aligned}$$

with two sorts $S = \{Nat, NatList\}$, where the types of function symbols are as follows:

$$\begin{aligned} nil &: NatList \\ 0 &: Nat \\ s &: Nat \to Nat \\ \text{“:”} &: Nat, NatList \to NatList \\ from &: Nat \to NatList \\ app &: NatList, NatList \to NatList \\ take &: Nat, NatList \to Nat \end{aligned}$$

If one restricts rewriting in $\mathcal{R}$ via Π given by

$$\langle x : from(y), 2, h\rangle \qquad \langle x : app(from(y), zs), 2.1, h\rangle$$
$$\langle x : app(y : app(z, zs), us), 2, h\rangle,$$

$\rightarrow_\Pi$ is terminating and still every well-formed ground term can be normalized with the restricted relation $\rightarrow_\Pi$ (provided the term is normalizing). See Examples 67 and 45 below for justifications of these claims.

Note that the pleasant behaviour (namely termination and normalization) of rewriting with forbidden pattern restrictions in Example 34 could neither be achieved with context-sensitive rewriting nor with on-demand rewriting or lazy rewriting, nor by using any reduction strategy.

Proposition 5. *Consider the TRS of Example 34.*

1. *It is non-terminating (even in the sorted version) under any reduction strategy.*

2. *Using context-sensitive, on-demand or lazy rewriting restrictions, the resulting restricted rewrite system is either non-terminating or original normal forms cannot be computed even if they exist.*

Proof. AD 1: Consider the (well-formed ground) term $\mathit{from}(0)$ and the infinite rewrite sequence

$$\mathit{from}(0) \rightarrow 0 : \mathit{from}(s(0)) \rightarrow 0 : s(0) : \mathit{from}(s(s(0))) \rightarrow \dots \tag{4.1}$$

Each term in this rewrite sequence contains exactly one redex, hence this reduction sequence respects every possible reduction strategy.

AD 2: For context-sensitive rewriting and lazy term rewriting a replacement map specifies the replacing/eager arguments of function symbols. Consider the function symbol ":". If $2 \in \mu(:)$, then the infinite reduction sequence 4.1 is a proper infinite context-sensitive resp. lazy rewrite sequence (in the case of lazy term rewriting for appropriately labeled terms and in particular, a canonically labelled start term; cf. [56, 77]). On the other hand, if $2 \notin \mu(:)$ the normal form of the (well-formed ground) term $\mathit{app}(0 : \mathit{nil}, s(0) : \mathit{nil})$ (which is $0 : (s(0) : \mathit{nil})$) cannot be computed, since the intermediate term $0 : \mathit{app}(\mathit{nil}, s(0) : \mathit{nil})$ is irreducible in both context-sensitive rewriting and lazy term rewriting (cf. [56, 77] for more details on and exact definitions of lazy term rewriting).

Finally, we consider on-demand rewriting. There we have two replacement maps μ and μ' where μ identifies the eager and μ' the lazy arguments of function symbols. If $i \notin \mu(f) \cup \mu'(f)$, then the i^{th} argument of f may under no circumstances be reduced. First, if $2 \in \mu(:)$ the infinite reduction sequence 4.1 is a proper on-demand reduction sequence and hence we have non-termination. On the other hand, if $2 \notin \mu(f)$, then (regardless of whether $2 \in \mu'(f)$ or not) the normal form of $\mathit{app}(0 : \mathit{nil}, s(0) : \mathit{nil})$ cannot be computed by on-demand rewriting analogous to the case of context-sensitive and lazy rewriting. □

4.2.2 Relations to Existing Approaches

Thanks to the generality of rewriting with forbidden patterns it is not surprising that several well-known approaches to restricted term rewriting as well as to rewriting guided by reduction strategies occur as special cases of rewriting with forbidden patterns. In the following we provide some examples. Context-sensitive rewriting w.r.t. to a replacement map μ arises as special case of rewriting with forbidden patterns when defining Π to contain for each function symbol f and each $j \in \{1, \dots, ar(f)\} \setminus \mu(f)$ the forbidden patterns $(f(x_1, \dots, x_{ar(f)}), j, h)$ and $(f(x_1, \dots, x_{ar(f)}), j, b)$.

Moreover, with forbidden patterns it is also possible to simulate position-based reduction strategies such as innermost and outermost rewriting. The innermost reduction relation of a TRS $\mathcal{R}$ coincides with the forbidden pattern reduction relation if one uses the forbidden patterns $\langle l, \epsilon, a\rangle$ for the left-hand sides l of each rule of $\mathcal{R}$. Dually, if patterns (l, ϵ, b) are used, the forbidden pattern reduction relation coincides with the *outermost* reduction relation w.r.t. $\mathcal{R}$.

On the other hand, more complex layered combinations of the aforementioned approaches, such as innermost context-sensitive rewriting cannot be modeled by forbidden patterns as proposed in this thesis.

Observation 2. *Innermost context-sensitive rewriting can in general not be modeled by rewriting with (a finite set of) forbidden patterns.*

Proof. Consider a rewrite system

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

over a signature $\mathcal{F} = \{a, f, g\}$ where a is a constant and f and g are unary function symbols. Moreover, let $\mu(f) = \{1\}$ and $\mu(g) = \emptyset$. Consider any finite set Π of forbidden patterns and let n be the maximal term depth of any forbidden pattern. Then consider a term $s = f^{n+1}(a)$ ($f^0(t) = t$ and $f^{m+1}(t) = f(f^m(t))$). The term s is not an innermost context-sensitive redex, thus if Π accurately models innermost context-sensitive rewriting, the root position of s must be forbidden by some forbidden pattern $\pi = \langle t, p, \lambda\rangle$. We have one of two possible situations.

First, assume t matches s, and either $p = \epsilon$ and $\lambda = h$, or $p > \epsilon$ and $\lambda = a$. Since the term depth of t is less or equal to n, $t = f^k(x)$ for some $k < n$. Hence, π also forbids the root position of the term $s' = f^{n+1}(x)$. However, s' is an innermost context-sensitive redex, thus in this case Π is not accurate in simulating innermost context-sensitive rewriting.

Second, assume t matches some proper subterm $s|_q$ of s and $\lambda = a$ (p is irrelevant in this case). Then, π forbids reductions at the root position of the term $f(g(s))$ because it matches $s|_q$. However, the term $f(g(s))$ is an innermost context-sensitive redex, thus also in this case Π is not accurate in simulating innermost context-sensitive rewriting. □

However, the generality and parameterizability of rewriting with forbidden patterns comes at a certain price. In order to make the approach feasible in practice, it is necessary to identify interesting classes of forbidden patterns that yield a reasonable trade-off between power and simplicity. For these interesting classes of forbidden patterns we need methods which guarantee that the results (e.g. normal forms) computed by rewriting with forbidden patterns are meaningful, in the sense that they have some natural correlation with the actual results obtained by unrestricted rewriting. For instance, it is desirable that normal forms w.r.t. the restricted rewrite system are original head-normal forms. In this case one can use the restricted reduction relation to compute original normal forms (by an iterated process) whenever they exist (provided that the TRS in question is left-linear, confluent and the restricted reduction relation is terminating) (cf. Section 4.5 below for details). Before presenting sufficient criteria for these "completeness" properties, we investigate the confluence and termination properties of rewrite systems with forbidden patterns.

4.3 Confluence of Rewriting with Forbidden Patterns

Typically, confluence of a restricted rewrite relation is of minor importance as compared to termination and the power to simulate unrestricted rewriting. The reason for that is the fact that restricted rewriting often computes certain "parts" of the actual result values, i.e. normal forms of terms. An example of this can be found e.g. in [55, Section 5.3] and also in Section 4.5 below. In both cases normal forms of the restricted rewrite relation are head-normal forms of the underlying unrestricted rewrite relation. Hence, restricted rewriting can be used to normalize terms (w.r.t. unrestricted rewriting) in an iterated, recursive process ([55, Theorem 23]). However, despite this close connection of unrestricted and restricted rewriting, the restricted rewrite relation might not be confluent even if the unrestricted one is. We provide an example from the realm of context-sensitive rewriting to illustrate this claim (note that this immediately yields an example for the same situation in rewriting with forbidden patterns, since context-sensitive rewriting is a special case of rewriting with forbidden patterns).

Example 35. *Consider the left-linear TRS $\mathcal{R} = (\mathcal{F}, R)$ from Example 34. It is non-terminating but confluent. When using a replacement map $\mu(:) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for $f \in \mathcal{F} \setminus \{:\}$, μ is less restrictive than the canonical replacement map for $\mathcal{R}$ and $\mathcal{R}$ is μ-terminating. Moreover, μ-normal forms are head-normal forms w.r.t. unrestricted rewriting according to [55, Theorem 8]. Hence, for a given normalizing term s its unique normal form (w.r.t. to unrestricted $\mathcal{R}$) can be computed by computing some μ-normal form t and recursively μ-normalize the arguments of $root(t)$. Nevertheless μ-normal forms are not unique even for* *normalizing terms:*

$$\begin{tikzcd}[column sep=small] app(0 : nil, app(nil, nil)) \arrow[r, "\mu"'] \arrow[d, "\mu"] & app(0 : nil, nil) \arrow[r, "\mu"'] & 0 : app(nil, nil) \\ 0 : app(nil, app(nil, nil)) & & \end{tikzcd}$$

However, despite the limited practical relevance of confluence of restricted rewriting, from a systematic point of view confluence criteria are nice to have and may provide an improved insight in the subtle technical details of rewriting with forbidden patterns.

4.3.1 Confluence by Local Confluence

It is well-known that for terminating rewrite relations local confluence implies confluence (Newman's Lemma [68]). Moreover, for ordinary terminating TRSs local confluence is decidable via the critical pair lemma of [45]. In this section we are going to develop a similar confluence criterion for left-linear TRSs with forbidden patterns. The main obstacle in the design of such a criterion is that local variable overlaps may not be joinable due to the forbidden pattern restrictions (cf. also [55, Example 7] for an example illustrating the analogous problem for context-sensitive rewriting).

Example 36. *Consider the following TRS $\mathcal{R}$.*

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

If one uses a set of forbidden patterns $\Pi = \{\langle g(x), 1, h\rangle\}$, $\to_\Pi$ is not locally confluent (despite $\to$ being confluent and terminating):

$$g(a) \leftarrow_\Pi f(a) \to_\Pi f(b)$$

The term $g(a)$ is a Π-normal form and obviously $f(b) \not\to_\Pi^* g(a)$. Hence, $\to_\Pi$ is not locally confluent.

The problem in Example 36 is that the redex a changes its status (from allowed to forbidden) in the transition from $f(a)$ to $g(a)$ although the term a is bound to a variable in the necessary rewrite step and thus only copied but not modified.

In order to avoid this problem we are going to impose restrictions on the forbidden patterns such that critical divergences as the one in Example 36 cannot occur. A similar approach was taken in [55] for context-sensitive rewriting. There, constraints on the status of variables occurring in rewrite rules were imposed in order to guarantee joinability of divergences obtained by variable overlaps. It is not surprising that the situation is more involved in the realm of rewriting with forbidden patterns, since the restrictions on the rewrite relations are much more fine-grained

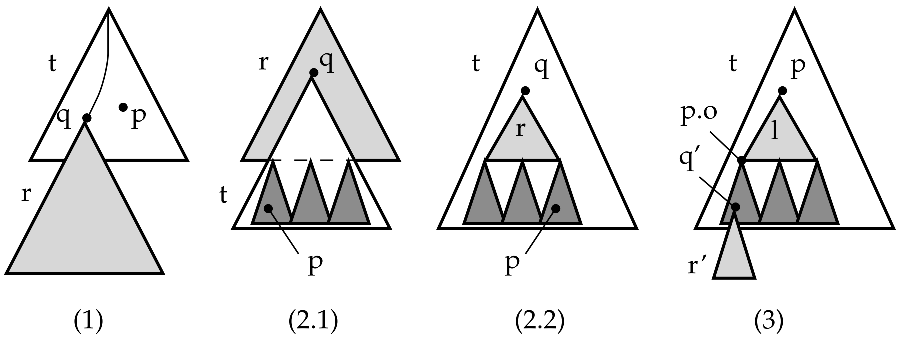


Figure 4.1: Critical h-patterns

there. We develop a confluence criterion for rewriting with forbidden patterns in the presence of h- and b-patterns. The reason we focus on these kind of patterns is that the termination methods of Section 4.4 are restricted to h- and b-patterns and the confluence criterion relies on termination. For simplicity reasons we define "non-criticality" of forbidden patterns separately for the cases of h- and b-patterns.

Definition 19 (non-critical forbidden h-patterns)**.** *Let $\mathcal{R}$ be a TRS and Π be a set of linear forbidden h-patterns. Π is said to be* non-critical *for $\mathcal{R}$ if for every $\pi = \langle t, p, h\rangle \in \Pi$ there is no rule $l \to r \in \mathcal{R}$ with*

1. *$t|_q\theta = r\theta$ for some position $q \in Pos_{\mathcal{F}}(t)$ and mgu θ where additionally $p \parallel q$;*

2. *$r|_q\theta = t\theta$ (or $r\theta = t|_q\theta$) for some $q \in Pos_{\mathcal{F}}(r)$ (resp. $q \in Pos_{\mathcal{F}}(t)$) and mgu θ where $p \geq o$ for some variable position $q.o$ of $r|_q$ (resp. $p \geq q.o$ for some variable position o of r);*

3. *$t|_p\theta = l\theta$ for some mgu θ and $t|_{q'}\theta' = r'\theta$ for some rule $l' \to r' \in \mathcal{R}$, mgu θ' and position $q' \geq p.o$ where $o \in Pos_{\mathcal{V}}(l)$.*

Figure 4.1 illustrates the cases of critical forbidden patterns described in Definition 19.

In the case of b-patterns we forbid any overlaps of forbidden patterns and right-hand sides of rules whatsoever, because in general all such overlaps lead to non-joinable local divergences w.r.t. rewriting with forbidden patterns for non-pathological TRSs.

Definition 20 (non-critical forbidden b-patterns)**.** *Let $\mathcal{R}$ be a TRS and Π be a set of linear forbidden b-patterns. Π is said to be* non-critical *for $\mathcal{R}$ if for every $\pi = \langle t, p, b\rangle \in \Pi$ there is no rule $l \to r \in \mathcal{R}$ such that $t|_q\theta = r\theta$ or $t\theta = r|_{q'}\theta$ for any positions $q \in Pos_{\mathcal{F}}(t)$ and $q' \in Pos_{\mathcal{F}}(r)$ and substitution θ.*

Definition 21 (non-critical forbidden patterns)**.** *Let $\mathcal{R}$ be a TRS and Π be a set of forbidden patterns. Π is* non-critical *for $\mathcal{R}$ if for all patterns $\pi = \langle t, p, \lambda\rangle \in \Pi$ we* *have that t is linear and $\lambda \in \{h, b\}$ and additionally Π_h and Π_b are both non-critical for $\mathcal{R}$, where Π_h is the subset of h-patterns of Π and Π_b is the subset of b-patterns of Π.*

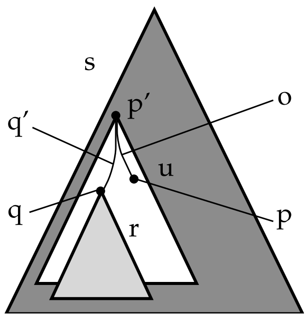
We sometimes just say that a set of forbidden patterns Π is non-critical, if the TRS $\mathcal{R}$ for which Π is non-critical is clear from the context. In the presence of non-critical forbidden patterns modifying a term at position p does not change the status of positions $q \parallel p$. Note that the definition of non-criticality for h- and b-patterns is only sufficient but not necessary in order to guarantee joinability of local divergences not corresponding to critical pairs (cf. Lemma 9 below). In particular, the context-sensitive notion of left-homogeneous replacing variables (cf. [55][Definition 5]) does not arise as special case of non-critical forbidden patterns. This suggests that generalizations of non-critical forbidden patterns are possible that still imply joinability of variable overlaps. Such generalizations are an interesting direction of future research in the area of confluence of rewriting with forbidden patterns.

The following lemma shows that the status of positions in a term is not altered from allowed to forbidden by reduction parallel to the position.

Proposition 6. *Let $\mathcal{R}$ be a TRS and let Π be a set of linear forbidden h- and b-patterns that are non-critical for $\mathcal{R}$. If position p is allowed in a term t, then it is also allowed in any term s obtained from t by a one-step reduction at some position $q \parallel p$.*

Proof. Assume to the contrary that p is forbidden in s. Hence, there is a forbidden pattern $\langle u, o, \lambda \rangle$ such that $s|_{p'} = u\sigma$ and $p'.o = p$ if $\lambda = h$ resp. $p'.o < p$ if $\lambda = b$.

Since u does not match $t|_{p'}$ (otherwise p would be forbidden in t) and is linear, it must be overlapped by the right-hand side of the rule applied in the step from t to s at position q' given by $p'.q' = q$. In case $\lambda = b$ we get an immediate contradiction to non-criticality of Π, since in that case a b-pattern is overlapped by the right-hand side of a rewrite rule (cf. Definition 20). In case $\lambda = h$ we have the following situation:



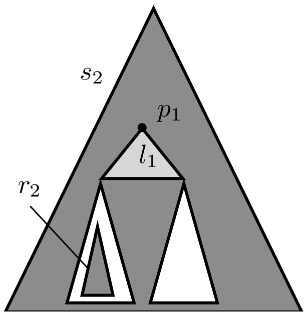
Since $p \parallel q$, we also have $q' \parallel o$ thus contradicting non-criticality of Π (precisely Item 1 of Definition 19). □

In systems restricted by non-critical forbidden patterns variable overlaps are non-critical.

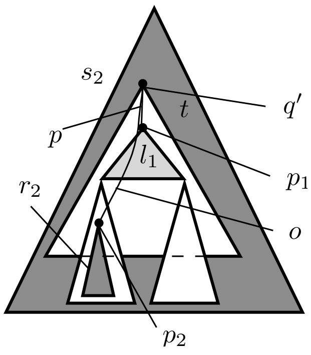
Lemma 9. *Let $\mathcal{R}$ be a left-linear TRS and Π be a set of linear forbidden h- and b-patterns that are non-critical for $\mathcal{R}$. Each local divergence $s_1 \xleftarrow{p_1}_{l_1 \to r_2, \Pi} s \xrightarrow{p_2}_{l_2 \to r_2, \Pi} s_2$ where either $p_1 \parallel p_2$ or $p_2 \geq p_1.q$ and $q \in Pos_V(l_1)$ is joinable by at most one parallel reduction step (on each side).*

Proof. First assume $p_1 \parallel p_2$. Thus, it suffices to show that p_1 is allowed in s_2 and p_2 is allowed in s_1. This is a direct consequence of Proposition 6.

Second, assume $p_2 = p_1.q.o$ for some $q \in Pos_V(l_1)$. Since $\mathcal{R}$ is left-linear, it suffices to show that p_1 is allowed in s_2 and all one-step descendants of p_2 are allowed in s_1. We consider position p_1 in s_2 first. Graphically the situation is the following:

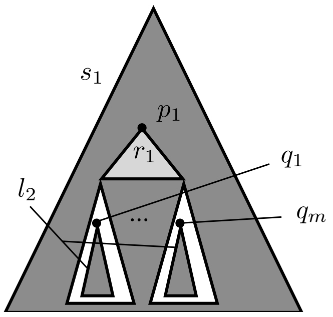


If position p_1 is forbidden in s_2, then there must be some pattern $\langle t, p, \lambda \rangle$ matching $s_2|_{q'}$ at some position $q' \leq p_1$ (say $q'.q'' = p_1$) such that either $q'.p = p_1$ if $\lambda = h$ or $q'.p < p_1$ if $\lambda = b$. For the latter case the situation is as follows.

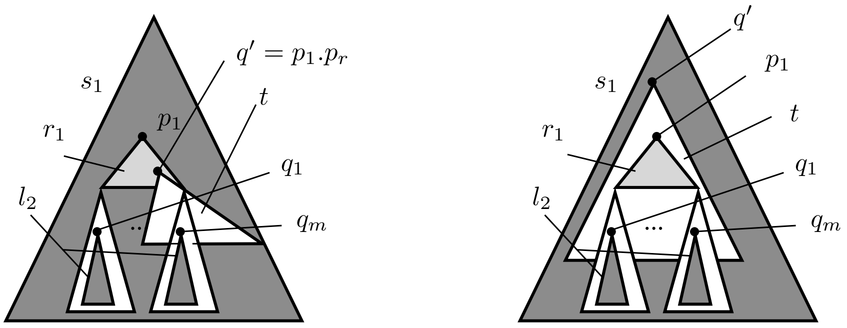


Since t does not match $s|_{q'}$ and t is linear, there must be some position $o \in Pos_{\mathcal{F}}(t)$ such that $t|_o$ and r_2 unify. Hence, if $\lambda = b$ we get an immediate contradiction to non-criticality of Π. Otherwise, we also get $t|_{q''}\theta = l_1\theta$ for some mgu θ, since $s_2|_{p_1}$ is already a common instance of $t|_{q''}$ and l_1. Moreover, $o \geq q''.p_V$ for some variable position of l_1, because $p_2 \geq p_1.p_V$. Hence, if $\lambda = h$ we get a contradiction by Item 3 of Definition 19.

Finally, we show that the descendants $\{q_1, \ldots, q_m\}$ of position p_2 are allowed in s_1. The situation is the following:



Assume some descendant q_i is forbidden in s_1. Then, there is some forbidden pattern $\pi = \langle t, p, \lambda \rangle$ matching $s_1|_{q'}$ at some position $q' \leq q_i$, thereby forbidding q_i. Since π does not forbid the reduction of $s|_{p_2}$, q' must either be of the shape $p_1.p_r$ for some $p_r \in Pos_{\mathcal{F}}(r_2)$ or a prefix of p_1 (i.e. $q' \leq p_1$):



In the latter case there must be some position $p_t \in Pos_{\mathcal{F}}(t)$ with $q'.p_t \geq p_1$, since otherwise t would have matched $s|_{q'}$ as t is linear. Hence, we have either $r_1\theta = t|_o\theta$ or $r_1|_o\theta = t\theta$ for some $o \in Pos_{\mathcal{F}}(t)$ (resp. $o \in Pos_{\mathcal{F}}(r_1)$) and mgu θ. Thus, in case $\lambda = b$ we have a contradiction to non-criticality of Π.

If $\lambda = h$, then in the first case we have $r_1|_{p_r}\theta = t\theta$ for some θ and $p \geq p_r.o$ for some variable position o of $r|_{p_r}$, because $q_i \geq p_1.o'$ for some variable position o' of r_1. In the second case we have $t|_{p_t}\theta = r_1\theta$ for some p_t (more precisely p_t is given by $q'.p_t = p_1$) and $p \geq p_t.o'$ for some variable position o' of r_1. In both cases we get a contradiction to Π being non-critical, more precisely to Item (2) of Definition 19. $\Box$

Example 37. *The forbidden patterns used in Example 33 are non-critical, while the forbidden patterns in (the untyped version of) Example 34 are critical, since the right-hand side $r = x : app(xs, ys)$ and t of the forbidden pattern $\langle t, p, h \rangle = \langle x' : app(from(y'), zs'), 2.1, h \rangle$ unify (indeed r matches t). In t the forbidden position is $p = 2.1$. This position is a variable position in r. Hence, we have a violation of Item 2 of Definition 19 (more precisely there we have $q = \epsilon$ and $p = o = 2.1$).*

Indeed $\rightarrow_{\mathcal{R},\Pi}$ is not (locally) confluent despite $\mathcal{R}$ being non-overlapping: To see *this, consider the local divergence $s_1 \leftarrow_\Pi s \rightarrow_\Pi s_2$ where*

$$\begin{array}{rcl} s &=& 0 : app(take(0, from(0) : nil), 0) \\ s_1 &=& 0 : app(from(0), 0) \\ s_2 &=& 0 : app(take(0, (0 : from(s(x))) : nil), 0) \end{array}$$

The term s_1 is a Π-normal form and s_2 can be further reduced (in a unique way):

$$\begin{array}{rcl} s_2 &=& 0 : app(take(0, (0 : from(s(x))) : nil), 0) \\ &\rightarrow_\Pi& 0 : app(0 : from(s(x), 0) \\ &\rightarrow_\Pi& 0 : (0 : app(from(s(x), 0)) \;=\; \widetilde{s_2} \end{array}$$

Since $\widetilde{s_2}$ is a normal form, $\widetilde{s_2} \neq s_1$ and the reduction sequence from s_2 to $\widetilde{s_2}$ was unique, s_1 and s_2 are not joinable.

Note that the term s is not well-typed, since $from(0)$ which is of type $NatList$ occurs in the first argument of ":" which demands a term of type Nat. We will later see that divergences starting from (normalizing) well-typed terms are joinable for this example (this is a consequence of Example 67 below and the fact that the unrestricted TRS is orthogonal and thus confluent).

One subtlety of rewriting with forbidden patterns is that parallel rewriting can in general not be simulated by multistep non-parallel rewriting (i.e. $\dashrightarrow\!\!\!\!\!\!+\!\!\!\!\!\!\!\!|\!|_\Pi \not\subseteq \rightarrow_\Pi^*$).

Example 38. *Consider the TRS $\mathcal{R} = (\mathcal{F}, R)$ where $\mathcal{F} = \{a, b, f\}$, a and b are constants, f is a function symbol of arity 2 and R is given by*

$$a \rightarrow b.$$

Additionally, consider a set of forbidden patterns $\Pi = \{\langle f(b, x), 2, h\rangle, \langle f(x, b), 1, h\rangle\}$. Then we have

$$f(a, a) \mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_\Pi f(b, b) \qquad but \qquad f(a, a) \not\rightarrow_\Pi^* f(b, b),$$

since both $f(a, b)$ and $f(b, a)$ are Π-normal forms. Hence $\mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_\Pi \not\subseteq \rightarrow_\Pi^*$.

In Example 38 the forbidden patterns Π are critical for the TRS $\mathcal{R}$ (because of Item 1 of Definition 19). Indeed, it turns out that $\mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_{\mathcal{R},\Pi} \subseteq \rightarrow_{\mathcal{R},\Pi}^*$ holds provided that Π is non-critical for $\mathcal{R}$.

Lemma 10. *Let $\mathcal{R}$ be a TRS and Π a set of linear non-critical forbidden h- and b-patterns. Then $\rightarrow_\Pi \subseteq \mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_\Pi \subseteq \rightarrow_\Pi^*$.*

Proof. The inclusion $\rightarrow_\Pi \subseteq \mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_\Pi$ is trivial, since every single step is by definition also a single parallel step. The second inclusion ($\mathrel{+\!\!\!\!\!|\!|\!\!\!\rightarrow}_\Pi \subseteq \rightarrow_\Pi^*$) is a straightforward consequence of Proposition 6. $\square$

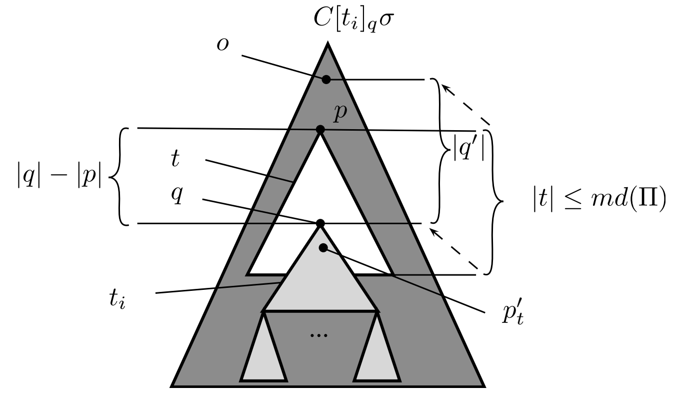
Building upon Lemma 9 we now provide a sufficient criterion of confluence of left-linear Π-terminating rewrite systems with non-critical forbidden patterns. To this end we are going to prove a modified critical pair lemma for rewriting with forbidden patterns. First we define the notion of rewrite sequences that are stable w.r.t. substitutions and contexts.

Definition 22. *Let $\rightarrow_\Pi$ be a forbidden pattern rewrite relation and t_i $(1 \leq i \leq n)$ a series of terms such that $t_1 \xrightarrow{p_1}_\Pi t_2 \xrightarrow{p_2}_\Pi \dots \xrightarrow{p_{n-1}}_\Pi t_n$. This reduction sequence is called stable w.r.t. contexts and substitutions (for $\rightarrow_\Pi$) if $C[t_1]_q\sigma \xrightarrow{q.p_1}_\Pi C[t_2]_q\sigma \xrightarrow{q.p_2}_\Pi \dots \xrightarrow{q.p_1}_\Pi C[t_n]_q\sigma$ holds for every substitution σ and every context $C[]_q$ where q is not forbidden in $C[]_q$.*

Proposition 7. *For a given TRS $\mathcal{R}$ and a set of linear forbidden h- and b-patterns it is decidable whether a given reduction sequence $t_1 \rightarrow_\Pi t_2 \rightarrow_\Pi \dots \rightarrow_\Pi t_n$ is stable under contexts and substitutions (for $\rightarrow_\Pi$).*

Proof. Let S be the reduction sequence $t_1 \rightarrow_\Pi t_2 \rightarrow_\Pi \dots \rightarrow_\Pi t_n$. Assume there is some substitution σ and context $C[]_q$ such that $C[t_1]_q\sigma \xrightarrow{q.p_1}_\Pi C[t_2]_q\sigma \xrightarrow{q.p_2}_\Pi \dots \xrightarrow{q.p_{n-1}}_\Pi C[t_n]_q\sigma$ does not hold (i.e. S is not stable under contexts and substitutions). We call C and σ witnesses for non-stability of S. This means that some reduction step is forbidden by Π, since the rewrite relation without restriction is stable under contexts and substitutions. As we are considering only linear h- and b-patterns, this means there is some forbidden pattern $\pi = \langle t, p_t, \lambda \rangle$ such that $C[t_i]_q\sigma|_p = t\tau$ for some substitution τ, some $1 \leq i < n$ and $p \in Pos_\mathcal{F}(C[t_i]_q)$, because otherwise π could not forbid a reduction at position $q.p_i$ where $p_i \in Pos_\mathcal{F}(t_i)$.

We are going to prove that there exist a context $C'[]_{q'}$ and a substitution σ' such that position $q'.p_i$ is forbidden in $C'[t_i]_{q'}\sigma'$ and the term depth of $C'[]_{q'}$ and $x\sigma'$ is at most equal to the largest term depth of any forbidden pattern in Π (denoted $md(\Pi)$ for maximal depth) for every $x \in Dom(\sigma')$: First, as q is not forbidden in $C[]_q$ we have $|q| - |p| \leq |t| \leq md(\Pi)$. Hence, we define the context $C'[]_{q'}$ by $C[]_q|_o$ where $q = o.q'$ and $|q'| = md(\Pi)$. Moreover, let p' be given by $p = o.p'$ (note that $p \geq o$, because there must be some position $p'_t \in Pos_\mathcal{F}(t)$ with $p.p'_t \geq q = o.q'$ as otherwise position q would be forbidden in $C[]_q$ as $|p'_t| \leq |q'| = md(\Pi)$).



Second, let θ be an mgu of $C'[t_i]_{q'}|_{p'}$ and t (which exists because we have $C'[t_i]_{q'}\sigma|_{p'} = t\tau$ and t_i and t are considered to be variable disjoint). We define σ' to be θ restricted to the variables of $Var(t_1)$ (note that $Var(t_1) \supseteq Var(t_i)$ for all $1 \leq i \leq n$) and thus $C'[t_i]_{q'}\sigma'|_{p'} = t\tau$ and $C'[t_i]_{q'}\sigma' \overset{q'.p_i}{\not\to}_{\Pi} C'[t_{i+1}]_{q'}\sigma'$. Since σ' is a restriction of the mgu θ and t is linear, $x\sigma'$ has a term depth of at most the term depth of t for every $x \in Dom(\sigma')$.

Hence, if S is not stable under contexts and substitutions, then we already find a witness context $C'[]_{p'}$ and substitution σ' for this non-stability in the finite set of contexts and substitutions involving only terms having a term depth not exceeding the maximal term depth of any forbidden pattern in Π. On the other hand, if S is stable under contexts and substitutions, then, in particular, there is no witness (context and substitution) to the contrary in this set of contexts and substitutions. Hence, stability of S under contexts and substitutions can be decided by checking whether S is stable under those contexts and substitutions that involve only terms having a term depth not exceeding the maximal term depth of any forbidden pattern in Π. □

In order to provide a modified critical pair lemma for rewriting with forbidden patterns we first define the notion of Π-critical pairs w.r.t forbidden patterns Π and a TRS $\mathcal{R}$, which are critical pairs that arise from an $\to_{\Pi}$ divergence.

Definition 23 (Π-critical pairs)**.** *Let $\mathcal{R}$ be a TRS and Π a set of forbidden patterns. Moreover, let $l_1 \to r_1$ and $l_2 \to r_2$ be rewrite rules from $\mathcal{R}$. If $l_1|_p\theta = l_2\theta$ for some mgu θ and some position $p \in Pos_{\mathcal{F}}(l_1)$ which is allowed in l_1, then $\langle l_1[r_2]_p\theta, r_1\theta\rangle$ is a Π-critical pair.*

Lemma 11 (adapted Critical Pair Lemma for TRSs with forbidden patterns)**.** *Let $\mathcal{R}$ be a left-linear TRS and Π be a set of linear non-critical forbidden h- and b-patterns. If all Π-critical pairs $\langle s,t\rangle$ of $\mathcal{R}$ are joinable through $\to_{\Pi}$ reductions that are stable under contexts and substitutions, then $\to_{\Pi}$ is locally confluent.*

Proof. Consider a local divergence

$$s_1 \overset{p_1}{\leftarrow}_{l_1\to r_1,\Pi}\, s \overset{p_2}{\to}_{l_2\to r_2,\Pi}\, s_2.$$

If $p_1 \parallel p_2$ we have $s_1 \downarrow_{\Pi} s_2$ according to Lemmata 9 and 10. Otherwise, assume w.l.o.g. that $p_1 \leq p_2$. If $p_1.p \leq p_2$ for some position $p \in Pos_V(l_1)$ we have $s_1 \downarrow_{\Pi} s_2$ by Lemmata 9 and 10 again. Otherwise, the divergence is due to a critical pair $\langle s,t\rangle$ and $s_1 = C[s]_q\sigma$ and $s_2 = C[t]_q\sigma$ for some context $C[]_q$ and some substitution σ. Since s and t are joinable with reductions that are stable under contexts and substitutions, we have $s_1 \downarrow_{\Pi} s_2$. □

Finally, we obtain a confluence criterion based on this adapted critical pair lemma.

Theorem 12. *Let $\mathcal{R}$ be a left-linear Π-terminating TRS and Π be a set of linear non-critical forbidden h- and b-patterns. If all π-critical pairs $\langle s, t\rangle$ of $\mathcal{R}$ are joinable through $\rightarrow_\Pi$ reductions that are stable under contexts and substitutions, then $\rightarrow_\Pi$ is confluent.*

Proof. Follows from Lemma 11 and Newman's Lemma ([68]), since $\mathcal{R}$ is Π-terminating. □

Example 39. *Consider the TRS $\mathcal{R}$ and forbidden patterns Π of Example 33. $\mathcal{R}$ has no critical pairs and thus no Π-critical pairs. Moreover, $\mathcal{R}$ is left-linear Π-terminating (cf. Example 44 below). Hence $\rightarrow_{\mathcal{R},\Pi}$ is confluent.*

4.3.2 Confluence by the Diamond Property

It is well known that the diamond property of any relation $\rightarrow$ implies confluence of this relation. Moreover, for any two relations $\rightarrow_1$ and $\rightarrow_2$ with $\rightarrow_1 \subseteq \rightarrow_2 \subseteq \rightarrow_1^*$ confluence coincides. These two results have been used to establish that left-linear non-overlapping (i.e. orthogonal) (unrestricted) rewrite systems are confluent. More precisely it can be shown that for orthogonal TRSs $\mathcal{R} \mathrel{+\!\!\!\!\!\!\!\!\rightarrow}_{\mathcal{R}}$ has the diamond property (this result is obtained from the so called *Parallel Moves Lemma*, cf. eg. [11, Lemma 6.4.4]) and as $\rightarrow_{\mathcal{R}} \subseteq \mathrel{+\!\!\!\!\!\!\!\!\rightarrow}_{\mathcal{R}} \subseteq \rightarrow_{\mathcal{R}}^*$ holds, $\rightarrow_{\mathcal{R}}$ is confluent.

Lemma 9 suggests that in the presence of non-critical forbidden patterns a modified version of the Parallel Moves Lemma might hold for rewriting with forbidden patterns. Indeed, this is the case for Π-orthogonal rewrite systems.

Definition 24 (Π-orthogonal)**.** *A rewrite system $\mathcal{R}$ with forbidden patterns Π is Π-orthogonal it $\mathcal{R}$ is left linear and there are no Π-critical pairs.*

Lemma 12 (adapted Parallel Moves Lemma for rewriting with forbidden patterns)**.** *Let $\mathcal{R}$ be a Π-orthogonal TRS and Π a set of linear non-critical forbidden h- and b-patterns. Then $\mathrel{+\!\!\!\!\!\!\!\!\rightarrow}_\Pi$ has the diamond property.*

Proof. First consider a local divergence

$$s_1 \xleftarrow{p}_\Pi s \overset{q_1,\dots,q_n}{\mathrel{+\!\!\!\!\!\!\!\!\rightarrow}}_\Pi s_2$$

where $p \leq q_i$ for all $1 \leq i \leq n$. Here $s_1 = s[r\sigma]_p$ (for some substitution σ and rewrite rule $l \rightarrow r$) and $s_2 = s[r_1\sigma_1]_{p_1} \dots [r_n\sigma_n]_{q_n}$ (for some substitutions $\sigma_1, \dots, \sigma_n$ and some rewrite rules $l_1 \rightarrow r_1, \dots, l_n \rightarrow r_n$).

We need to show that p is allowed in s_2 and all descendants of positions from $\{q_1, \dots, q_n\}$ are allowed in s_1. We denote this set of descendants by $Desc_{q_1,\dots,q_n}$. Regarding the status of positions from $Desc_{q_1,\dots,q_n}$ we already know by (the proof of) Lemma 9 that in the single divergences

$$s_1 \xleftarrow{p}_\Pi s \xrightarrow{q_i}_\Pi s_2^i$$

the descendants of q_i in s_1 are allowed for all $1 \leq i \leq n$.

Regarding the status of position p in s_2 we prove by induction on n that it is allowed. If $n = 1$, p is allowed according to (the proof of) Lemma 9. If $n > 1$, the induction hypothesis yields that p is allowed in s_2' obtained from s by $s \overset{q_1,\ldots,q_{n-1}}{\mathrel{-\!\!\Vert\!\!\rightarrow}}_{\Pi} s_2'$ using the rewrite rules $l_1 \rightarrow r_1, \ldots, l_{n-1} \rightarrow r_{n-1}$. Since $\mathcal{R}$ is Π-orthogonal, $s_2'|_p = l\sigma$ (for some σ) and moreover position q_n is allowed in s_2' according to Proposition 6. Hence, we have a divergence

$$s_1' \overset{p}{\leftarrow}_{\Pi} s_2' \overset{q_n}{\rightarrow}_{\Pi} s_2$$

for which again (the proof of) Lemma 9 yields that position p is allowed in s_2.

Now consider a general divergence

$$s_1 \overset{p_1,\ldots,p_m}{\mathrel{\leftarrow\!\!\Vert\!\!-}}_{\Pi} s \overset{q_1,\ldots,q_n}{\mathrel{-\!\!\Vert\!\!\rightarrow}}_{\Pi} s_2.$$

Let $P^{max} = \{p_1^{max}, \ldots, p_k^{max}\}$ be the set of maximal positions from $\{p_1, \ldots, p_m, q_1, \ldots, q_n\}$ We decompose the divergence into several divergences of the above shape:

$$s_1^i \overset{p_i^{max}}{\leftarrow}_{\Pi} s \overset{q_1^i,\ldots,q_{l_i}^i}{\mathrel{-\!\!\Vert\!\!\rightarrow}}_{\Pi} s_2^i$$

for all $1 \leq i \leq k$ where $\{q_1^i, \ldots, q_{l_i}^i\}$ is the subset of positions from $\{p_1, \ldots, p_m, q_1, \ldots, q_n\}$ that are greater than p_i^{max}. For these divergences we already proved that p_i^{max} is allowed in s_2^i for every $1 \leq i \leq k$ and the descendants of the positions $q_1^i, \ldots, q_{l_i}^i$ are allowed in s_1^i and below p_i^{max} for all $1 \leq i \leq k$. By definition positions in P^{max} are pairwise parallel. Hence, we can apply Proposition 6 to obtain that all positions from P^{max} are allowed in s_1 or s_2 (depending on whether these positions originally were from $\{p_1, \ldots, p_m\}$ or $\{q_1, \ldots, q_n\}$) and all descendents of the positions $q_1^i, \ldots, q_{l_i}^i$ are allowed in s_1 or s_2 (depending on whether these positions originally were from $\{p_1, \ldots, p_m\}$ or $\{q_1, \ldots, q_n\}$). Hence, as descendants of parallel positions are again parallel we have joinability of s_1 and s_2 with one $\mathrel{-\!\!\Vert\!\!\rightarrow}_{\Pi}$ step on each side. □

We obtain a confluence criterion based on Lemma 12 and Lemma 10:

Theorem 13. *Let $\mathcal{R}$ be a Π-orthogonal TRS where Π is a set of linear non-critical forbidden h- and b-patterns. Then $\rightarrow_{\Pi}$ is confluent.*

Proof. By Lemma 12, $\mathrel{-\!\!\Vert\!\!\rightarrow}_{\Pi}$ has the diamond property and thus is confluent ([11, Lemma 2.7.4]). Moreover, we have $\rightarrow_{\Pi} \subseteq \mathrel{-\!\!\Vert\!\!\rightarrow}_{\Pi} \subseteq \rightarrow_{\Pi}^{*}$ by Lemma 10 and thus obtain confluence of $\rightarrow_{\Pi}$ by [11, Lemma 2.7.7]. □

Example 40. *Consider the TRS $\mathcal{R}$ and the forbidden patterns Π of Example 33. Since $\mathcal{R}$ is Π-terminating and Π is non-critical for $\mathcal{R}$, we conclude confluence of $\rightarrow_{\mathcal{R},\Pi}$. Note that in contrast to Example 37 we did not use Π-termination of $\mathcal{R}$ to obtain confluence here.*

4.4 Termination of Rewriting with Forbidden Patterns

We describe two different approaches for the termination analysis of TRSs restricted by forbidden patterns. The first one is a transformative method applicable to TRSs with linear forbidden h-patterns. In this approach we exploit the fact that for this kind of patterns (say Π) the status of some position p in a term s is determined by the function symbols (or variables) of s at positions q with $||q|-|p|| < max_{\langle t,p,h\rangle \in \Pi}|t|$ and that this set of positions is finite.

The second approach is based on a modified dependency pair framework (cf. [33]) comparable to [2] which we also used in Section 3.2.4. The basic idea is to enrich dependency pairs by the contexts in which the corresponding recursive function calls take place in the right-hand sides of the rewrite rules. This additional contextual information is used to define forbidden pattern dependency pair chains which correspond to forbidden rewrite sequences such that infinite dependency pair chains exist if and only if infinite forbidden pattern rewrite chains exist. Thus, in order to prove forbidden pattern termination one proves the absence of infinite forbidden pattern dependency pair chains with (some of) the usual advantages one obtains by using the dependency pair framework.

4.4.1 Termination of Rewriting with Forbidden Patterns by Transformation

In this subsection we are exclusively concerned with linear h-patterns. For this kind of forbidden patterns (and assuming a finite signature) a (finite) set of allowed contexts and substitutions under which a rewrite rule may be applied complementing the forbidden ones can be constructed. Thus, we can transform a rewrite system with this kind of forbidden patterns into a standard (i.e., context-free) one by explicitly instantiating and embedding all rewrite rules (in a minimal way) in contexts (including a designated top-symbol representing the *empty* context) such that rewrite steps in these contexts and under these substitutions are allowed.

To this end we propose a transformation that proceeds by iteratively instantiating and embedding rules in a minimal way. This is to say that the used substitutions map variables only to terms of the form $f(x_1, \ldots, x_{ar(f)})$ and the contexts used for the embeddings have the form $g(x_1, \ldots, x_{i-1}, \Box, x_{i+1}, \ldots, x_{ar(f)})$ for some function symbols $f \in \mathcal{F}$, $g \in \mathcal{F} \uplus \{\mathsf{top}\}$ and some argument position i of f (resp. g). It is important to keep track of the position of the initial rule inside the embeddings. Thus we associate to each rule introduced by the transformation a position pointing to the embedded original rule. This is to say that if a rewrite rule $l \to r$ is embedded into a context $C[]_p$, thus yielding a rewrite rule $C[l]_p \to C[r]_p$, we associate the position p to this rewrite rule, which points to the location of the lhs resp. rhs of the original rule in the lhs resp. rhs of the embedded rule. We denote such *embedded rewrite rules* by pairs of rewrite rules and positions. For instance we write $\langle C[l]_p \to C[r]_p, p\rangle$. To all initial rules of $\mathcal{R}$ we associate ϵ.

Note that it is essential to consider a new unary function symbol top_s for every sort $s \in S$ (of type $s \to s$) representing the empty context. This is illustrated by the following example.

Example 41. *Consider the TRS given by*

$$a \to f(a) \qquad f(x) \to x$$

with $\mathcal{F} = \{a, f\}$ *and the set of forbidden patterns* $\Pi = \{\langle f(x), 1, h\rangle\}$. *This system is not* Π*-terminating as we have*

$$a \to_\Pi f(a) \to_\Pi a \to_\Pi \dots$$

Whether a subterm $s|_p = a$ *of some term* s *is allowed for reduction by* Π *depends on its context. Thus, according to the idea of our transformation we try to identify all contexts* $C[a]_p$ *such that the reduction of* a *at position* p *is allowed by* Π. *However, there is no such (non-empty) context, although* a *may be reduced if* C *is the empty context. Moreover, there cannot be a rule* $l \to r$ *in the transformed system where* $l = a$, *since that would allow the reduction of terms that might be forbidden by* Π. *Our solution to this problem is to introduce a new function symbol* top *explicitly representing the empty context. Thus, in the example the transformed system will contain a rule* $\mathsf{top}(a) \to \mathsf{top}(f(a))$.

Abusing notation we subsequently use only one top-symbol, while we actually mean the top_s-symbol of the appropriate sort. All forbidden patterns used in this section (particularly in the lemmata) are linear here-patterns. We will make this general assumption explicit only in the more important results.

The following definition introduces the function T that, given an embedded rewrite rule $\langle l \to r, p\rangle$, computes the set $\{\langle l_1 \to r_1, p_1\rangle, \dots, \langle l_n \to r_n, p_n\rangle\}$ of (minimally) embedded and instantiated versions of this rule, where additionally l_i overlaps with some forbidden pattern for all $1 \le i \le n$. The intuition behind demanding that the computed embedded and instantiated rewrite rules have to overlap with forbidden patterns is that in this case further embedding and instantiation of these embedded rules might lead to rules whose left-hand sides are actually matched by forbidden patterns.

Definition 25 (instantiation and embedding)**.** *Let* $\mathcal{F} = (S, \Omega)$ *be a signature, let* $\langle l \to r, p\rangle$ *be an embedded rewrite rule of sort* s *over* $\mathcal{F}$ *and let* Π *be a set of linear* h*-forbidden patterns. The set of minimal instantiated and embedded rewrite rules*

$T_\Pi(\langle l \to r, p\rangle)$ (or just $T(\langle l \to r, p\rangle)$) is $T^i_\Pi(\langle l \to r, p\rangle) \uplus T^e_\Pi(\langle l \to r, p\rangle)$[2] *where*

$$\begin{aligned}
T^e(\langle l \to r, p\rangle) &= \{\langle C[l] \to C[r], i.p\rangle \mid C = f(x_1, \ldots, x_{i-1}, \Box, x_{i+1}, \ldots, x_{ar(f)}),\\
&\quad f \in \Omega_{(s_1,\ldots,s_{i-1},s,s_{i+1},\ldots,s_{ar(f)}),s'}, f \in \mathcal{F} \uplus \{\mathsf{top}_s \mid s \in S\},\\
&\quad i \in \{1, \ldots, ar(f)\}, \exists\langle u, o, h\rangle \in \Pi\ \exists\theta.\ o = q.p \wedge q \neq \epsilon \wedge u|_q\theta = l\theta\}\\
T^i_\Pi(\langle l \to r, p\rangle) &= \{\langle l\sigma \to r\sigma, p\rangle \mid x\sigma = f(x_1, \ldots, x_{ar(f)}),\\
&\quad sort(x) = sort(f(x_1, \ldots x_{ar(f)})),\\
&\quad f \in \mathcal{F}, y \neq x \Rightarrow y\sigma = y, x \in RV_\Pi(l, p)\}
\end{aligned}$$

and $RV_\Pi(l, p) = \{x \in Var(l) \mid \exists\langle u, o, h\rangle \in \Pi.\theta = mgu(u, l|_q) \wedge q.o = p \wedge x\theta \notin V\}$. The variables $x_1, \ldots, x_{ar(f)}$ are fresh.

We also call the elements of $T(\langle l \to r, p\rangle)$ the one-step T-successors of $\langle l \to r, p\rangle$. The reflexive-transitive closure of the one-step T-successor relation is the many-step T-successor relation or just T-successor relation. We denote the set of all many-step T-successors of an embedded rule $\langle l \to r, p\rangle$ by $T^*(\langle l \to r, p\rangle)$.

The set $RV_\Pi(l, p)$ of "relevant variables" is relevant in the sense that their instantiation might contribute to a matching by some (part of a) forbidden pattern term.

Note that in the generated embedded rewrite rules $\langle l' \to r', p'\rangle$ in $T_\Pi(\langle l \to r, p\rangle)$, a fresh top_s-symbol can only occur at the root of both l' and r' or not at all, according to the construction in Definition 25.

Example 42. *Consider the TRS $(\mathcal{R}, \Pi)$ where $\mathcal{R} = (\{a, f, g\}, \{f(x) \to g(x)\})$ and the forbidden patterns Π are given by $\{\langle g(g(f(a))), 1.1, h\rangle\}$. $T(\langle f(x) \to g(x), \epsilon\rangle)$ consists of the following embedded rewrite rules.*

$$\begin{aligned}
\langle f(f(x)) &\to g(f(x)), \epsilon\rangle && (4.2)\\
\langle f(g(x)) &\to g(g(x)), \epsilon\rangle && (4.3)\\
\langle f(a) &\to g(a), \epsilon\rangle && (4.4)\\
\langle f(f(x)) &\to f(g(x)), 1\rangle && (4.5)\\
\langle g(f(x)) &\to g(g(x)), 1\rangle && (4.6)\\
\langle top(f(x)) &\to top(g(x)), 1\rangle && (4.7)
\end{aligned}$$

Note that $RV_\Pi(f(x), \epsilon) = \{x\}$, because $g(g(f(a)))_{1.1} = f(a)$ unifies with $f(x)$ and mgu θ where $x\theta = a \notin V$. On the other hand $RV_\Pi(f(f(x)), 1) = \emptyset$.

Lemma 13 (finiteness of instantiation and embedding)**.** *Let $\langle l \to r, p\rangle$ be an embedded rewrite rule and let Π be a set of forbidden patterns. The set of (many-step) instantiations and embeddings of $\langle l \to r, p\rangle$ (i.e. $T^*(\langle l \to r, p\rangle)$) is finite.*

[2] Note that $T^i_\Pi(\langle l \to r, p\rangle)$ and $T^e_\Pi(\langle l \to r, p\rangle)$ are disjoint, because the second component (i.e. the position) of embedded rewrite rules is p for all embedded rewrite rules in $T^i_\Pi(\langle l \to r, p\rangle)$, but not equal to p for all embedded rewrite rules in $T^e_\Pi(\langle l \to r, p\rangle)$.

Proof. Assume towards a contradiction that $T^*(\langle l \to r, p\rangle))$ were infinite. Each embedded rule from $T^*(\langle l \to r, p\rangle))$ has the form $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$ for some context C and some substitution σ. Now infinity of $T^*(\langle l \to r, p\rangle))$ implies that the term depth of its terms is not bounded. Thus, it either contains embedded rules $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$, where the term depth of $x\sigma$ is n for arbitrarily high n and some (fixed) x, or it contains embedded rules $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$ where $|q|$ is arbitrarily big.

We investigate both cases. First, assume there is some variable x such that the term depth of $x\sigma$ is not bounded in $T^*(\langle l \to r, p\rangle))$. Let $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$ be an embedded rule such that the term depth of $x\sigma = n$ for some $x \in Var(C[l]_q)$ where $n > max_{\langle u,o,h\rangle \in \Pi}(depth(u))$. As the term depth of $x\sigma$ is not bounded in $T^*(\langle l \to r, p\rangle))$, some embedded rule of this shape must have a one-step T-successor $\langle C[l]_q\sigma' \to C[r]_q\sigma', q.p\rangle$ with $depth(x\sigma') > depth(x\sigma)$. Say $\sigma' = \sigma\sigma''$, $y\sigma'' \neq y$ and $x\sigma|_r = y$ with $depth(r) = n$. Thus, according to Definition 25, $y \in RV_\Pi(C[l]_q, q.p)$. Therefore, $C[l]_q\sigma|_{q'}\theta = u\theta$ for some $\langle u,o,h\rangle \in \Pi$ with $q' \leq p.q$ and $y\theta \neq y$. However, because of linearity of u this means $u|_{p''}$ is non-variable where $p'' = q''.r$ for some q''. However this contradicts the fact that the term depth of u is smaller than $n = |r|$.

Second, assume we have embedded rules $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$ with arbitrarily high $|q|$. Let $\langle C[l]_q\sigma \to C[r]_q\sigma, q.p\rangle$ be an embedded rule such that $|q| = n$ where $n > max_{\langle u,o,h\rangle \in \Pi}(depth(u))$. Some embedded rule of this shape must have a one-step T-successor $\langle C[l]_{q'}\sigma \to C[r]_{q'}\sigma, q.p\rangle$ with $|q'| > |q|$. Thus, according to Definition 25 there is a forbidden pattern $\langle u,o,h\rangle \in \Pi$ such that $u|_{p'} = l$ and $o = p'.q.p$ which contradicts $|o| < |q|$. □

The transformation we are proposing proceeds by iteratedly instantiating and embedding rewrite rules. The following definitions identify the embedded rules for which no further instantiation and embedding is needed.

Definition 26 (Π-stable)**.** *Let $\langle l \to r, p\rangle$ be an embedded rewrite rule and let Π be a set of forbidden patterns. We say that $\langle l \to r, p\rangle$ is Π-stable ($stb_\Pi(\langle l \to r, p\rangle)$ for short) if there is no context C and no substitution σ such that $C[l\sigma]_q|_{q'} = u\theta$ and $q.p = q'.o$ for any forbidden pattern $\langle u,o,h\rangle \in \Pi$ and any θ.*

Note that Π-stability is effectively decidable (for finite signatures and finite Π), since only contexts and substitutions involving terms not exceeding a certain depth depending on Π need to be considered.

Definition 27 (Π-obsolete)**.** *Let $\langle l \to r, p\rangle$ be an embedded rewrite rule and let Π be a set of forbidden patterns. We say that $\langle l \to r, p\rangle$ is Π-obsolete ($obs_\Pi(\langle l \to r, p\rangle)$ for short) if there is a forbidden pattern $\Pi = \langle u,o,h\rangle$ such that $l|_q = u\theta$ and $p = q.o$.*

In Example 42, the rules (4.2), (4.3), (4.5) and (4.7) are Π-stable, while rules (4.4) and (4.6) would be processed further. After two more steps e.g. an embedded rule $\langle g(g(f(a))) \to g(g(g(a))), 1.1\rangle$ is produced that is Π-obsolete.

The following lemmata state some properties of Π-stable embedded rules.

Lemma 14. *Let Π be a set of forbidden patterns and let $\langle l' \to r', p\rangle$ be a Π-stable embedded rewrite rule where $l' = C[l\sigma]_p$ and $r' = C[r\sigma]_p$ for some rewrite rule $l \to r$. If $s \to t$ with $l' \to r'$, then $s \to_\Pi t$ with $l \to r$.*

Proof. Suppose $s = s[l'\theta]_q \to s[r'\theta]_q = t$. If $s|_{q.p}$ were forbidden for reduction by Π (say through a forbidden pattern $\langle u, o, h\rangle$), then $s|_{p'} = u\theta'$ and $q.p = p'.o$. This is a direct contradiction to the fact that $l' \to r'$ is Π-stable according to Definition 26. Hence, $s|_{q.p}$ is allowed according to Π and we have $s = s[l\theta']_{q.p} \to_\Pi s[r\theta']_{q.p} = t$. $\Box$

Lemma 15. *Let $\langle l \to r, p\rangle$ be an embedded rule and Π be a set of forbidden patterns. If $T(\langle l \to r, p\rangle) = \emptyset$, then $\langle l \to r, p\rangle$ is either Π-stable or Π-obsolete.*

Proof. Assume $\langle l \to r, p\rangle$ is neither Π-stable nor Π-obsolete. Then, there exist a context C and a substitution σ such that $C[l\sigma]_q|_{q'} = u\theta$ and $q.p = q'.o$ for some pattern $\langle u, o, h\rangle$ and on the other hand there is no pattern $\langle u, o, h\rangle$ such that $l|_q = u\theta$ and $p = q.o$. Hence, either C or σ is non-trivial (i.e. $C \neq \Box$ or $x\sigma \neq x$ for some x). Assume C is non-trivial, then there exists a pattern $\langle u, o, h\rangle$ with $u|_q\theta = l\theta$, $q \neq \epsilon$ and $o = q.p$, hence $T^e_\Pi(\langle l \to r, p\rangle) \neq \emptyset$ and we get a contradiction.

On the other hand, assume σ is non-trivial and C is trivial (say $x\sigma \neq x$). Then $x \in RV_\Pi(l, p)$ and thus $T^i_\Pi(\langle l \to r, p\rangle) \neq \emptyset$. Hence we get a contradiction as well. $\Box$

Definition 28. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS with an associated set of forbidden patterns Π where $\mathcal{F} = (S, \Omega)$. The transformation $\mathbb{T}$ maps TRSs with forbidden patterns to standard TRSs $\mathbb{T}(\mathcal{R}, \Pi)$. It proceeds iteratively using 5 basic steps.*

1. $R^{tmp} = \{\langle l \to r, \epsilon\rangle \mid l \to r \in R\}$
   $R^{acc} = \emptyset$
2. $R^{acc} = \{\langle l \to r, p\rangle \in R^{tmp} \mid stb_\Pi(\langle l \to r, p\rangle)\}$
   $R^{tmp} = \{\langle l \to r, p\rangle \in R^{tmp} \mid \neg stb_\Pi(\langle l \to r, p\rangle) \wedge \neg obs_\Pi(\langle l \to r, p\rangle)\}$
3. $R^{tmp} = \bigcup_{\langle l \to r, p\rangle \in R^{tmp}} T(\langle l \to r, p\rangle)$
4. *If $R^{tmp} \neq \emptyset$ go to 2*
5. $\mathbb{T}(\mathcal{R}, \Pi) = (\mathcal{F} \uplus \{top_s \mid s \in S\}, \{l \to r \mid \langle l \to r, p\rangle \in R^{acc}\})$

In the transformation embedded rewrite rules are iteratively created and collected in R^{tmp} (temporary rules). Those embedded rules that are Π-stable and will thus be present in the final transformed system are collected in R^{acc} (accepted rules).

Lemma 16. *Let $\mathcal{R}$ be a rewrite system and Π be a set of forbidden linear h-patterns. If $s \to_{\mathcal{R},\Pi} t$ for ground terms s and t, then $\mathsf{top}(s) \to \mathsf{top}(t)$ in $\mathbb{T}(\mathcal{R}, \Pi)$.*

Proof. Assume the step $s \to_\Pi t$ occurs at position p with rule $l \to r$. If $\langle l \to r, \epsilon\rangle$ is Π-stable, we have $\mathsf{top}(s) \to \mathsf{top}(t)$ with $l \to r$ at position $1.p$ and the claim holds. $\langle l \to r, \epsilon\rangle$ cannot be Π-obsolete, since this would contradict the fact that

p is allowed in s according to Π. Thus, according to Lemma 15, $T(\langle l \to r, \epsilon\rangle)$ is non-empty and thus in particular, contains an embedded rule $\langle l' \to r', p'\rangle$ such that $\mathsf{top}(s)|_q = l'\sigma$ and $1.p = q.p'$, since all possible instantiations and embeddings are covered by T (note that the rule is also embedded in the $\mathsf{top}(\Box)$-context). This embedded rule cannot be Π-obsolete, as this would imply that p is a forbidden position in s, because l' matches (a subterm of) $\mathsf{top}(s)$ and thus a forbidden pattern matching l' would also match s (note that forbidden pattern terms do not contain top). Hence, again either the embedded rule is Π-stable in which case we use it for the reduction $\mathsf{top}(s) \to_{T(\mathcal{R})} \mathsf{top}(t)$, or it is further instantiated and/or embedded. By repetition we obtain new sets of embedded rules each containing rules whose left-hand sides match $\mathsf{top}(s)$ and thus are not Π-obsolete. By Lemmata 15 and 13 eventually one of these rules must be Π-stable and thus be in $T(\mathcal{R}, \Pi)$. Hence, we finally get $\mathsf{top}(s) \to_{T(\mathcal{R},\Pi)} \mathsf{top}(t)$. $\Box$

Theorem 14. *Let $\mathcal{R}$ be a TRS and Π be a set of linear here-patterns. We have $s \to^+_\Pi t$ for ground terms s and t if and only if $\mathsf{top}(s) \to^+_{\mathbb{T}(\mathcal{R},\Pi)} \mathsf{top}(t)$.*

Proof. The result is a direct consequence of Lemmata 14 and 16. $\Box$

Corollary 7. *Let $\mathcal{R}$ be a TRS and Π be a set of linear h-patterns. $\mathcal{R}$ is ground terminating under Π if and only if $\mathbb{T}(\mathcal{R}, \Pi)$ is ground terminating.*

Note that the restriction to ground terms is crucial in Corollary 7. Moreover, ground termination and general termination do not coincide in general for rewrite systems with forbidden patterns (observe that the same is true for other important rewrite restrictions and strategies such as the outermost strategy).

Example 43. *Consider the TRS $\mathcal{R} = (\mathcal{F}, R)$ given by $\mathcal{F} = \{a, f\}$ (where a is a constant) and R consisting of the rule*

$$f(x) \quad\to\quad f(x).$$

Moreover, consider the set of forbidden patterns $\Pi = \{\langle f(a), \epsilon, h\rangle, \langle f(f(x)), \epsilon, h\rangle\}$. Then $\mathcal{R}$ is not Π-terminating, because we have $f(x) \to_\Pi f(x)$ but it is Π-terminating on all ground terms, as can be shown by Theorem 14, since $T(\mathcal{R}, \Pi) = \emptyset$.

Example 44. *Consider the TRS of Example 33. We use two sorts $NatList$ and Nat, with function symbol types $2nd\colon NatList \to Nat$, $inf\colon Nat \to NatList$, $\mathsf{top}\colon NatList \to NatList$ (note that another "top" symbol of type $Nat \to Nat$ is not needed here), $s\colon Nat \to Nat$, $0\colon Nat$, $nil\colon NatList$ and $:$ of type $Nat, NatList \to NatList$. According to Definition 28, the rules of $\mathbb{T}(\mathcal{R}, \Pi)$ are:*

$$\begin{array}{rcl}
2nd(inf(x)) & \to & 2nd(x : inf(s(x))) \\
2nd(x : (y : zs)) & \to & y \\
\mathsf{top}(inf(x)) & \to & \mathsf{top}(x : inf(s(x))) \\
2nd(x' : inf(x)) & \to & 2nd(x' : (x : inf(s(x)))) \\
\mathsf{top}(x' : inf(x)) & \to & \mathsf{top}(x' : (x : inf(s(x))))
\end{array}$$

This system is terminating (and termination can be verified automatically, e.g. by AProVE [31]). Hence, by Corollary 7 also the TRS with forbidden patterns from Example 33 is ground terminating.

Example 45. *The TRS $\mathcal{R}$ and the forbidden patterns Π from Example 34 yield the following system $\mathbb{T}(\mathcal{R},\Pi)$. For the sake of saving space we abbreviate app by a, take by t and inf by i.*

$$
\begin{aligned}
\mathsf{top}(i(x)) &\to \mathsf{top}(x : i(s(x))) & t(y, i(x)) &\to t(y, x : i(s(x)))\\
a(y, i(x)) &\to a(y, x : i(s(x))) & \mathsf{top}(a(i(x), y)) &\to \mathsf{top}(a(x : i(s(x)), y))\\
t(a(i(x), y), z) &\to t(a(x : i(s(x)), y), z) & t(z, a(i(x), y)) &\to t(z, a(x : i(s(x)), y))\\
a(a(i(x), y), z) &\to a(a(x : i(s(x)), y), z) & a(z, a(i(x), y)) &\to a(z, a(x : i(s(x)), y))\\
\mathsf{top}(a(x : xs, ys)) &\to \mathsf{top}(x : a(xs, ys)) & t(z, a(x : xs, ys)) &\to t(z, x : a(xs, ys))\\
a(a(x : xs, ys), z) &\to a(x : a(xs, ys), z) & a(z, a(x : xs, ys)) &\to a(z, x : a(xs, ys))\\
a(x : i(zs), ys) &\to x : a(i(zs), ys) & a(x : s(zs), ys) &\to x : a(s(zs), ys)\\
a(x : (y : zs), ys) &\to x : a(y : zs, ys) & a(nil, x) &\to x\\
t(s(x), y : ys) &\to t(x, ys) & t(0, y : ys) &\to y\\
t(x, nil) &\to 0
\end{aligned}
$$

This system is terminating (and termination can be verified automatically, e.g. by AProVE [31]). Hence, again by Corollary 7 also the TRS with forbidden patterns from Example 34 is ground terminating.

4.4.1.1 Proving Termination of TRSs Obtained by $\mathbb{T}$

One major practical drawback of the transformation $\mathbb{T}$ is that the natural partition of function symbols into defined symbols and constructors is not preserved in general.

Example 46. *Consider the (untyped) TRS $\mathcal{R}$ and forbidden patterns Π of Example 32. $\mathbb{T}(\mathcal{R},\Pi)$ is given by the following rewrite rules.*

$$
\begin{aligned}
2nd(inf(x)) &\to 2nd(x : inf(s(x)))\\
s(inf(x)) &\to s(x : inf(s(x)))\\
inf(x) : y &\to (x : inf(s(x))) : y\\
top(inf(x)) &\to top(x : inf(s(x)))\\
s(x' : inf(x)) &\to s(x' : x : inf(s(x)))\\
top(x' : inf(x)) &\to top(x' : x : inf(s(x)))\\
2nd(x : y : zs) &\to y\\
inf(inf(x)) &\to inf(x : inf(s(x)))\\
(x' : inf(x)) : y &\to (x' : (x : inf(s(x)))) : y\\
2nd(x' : inf(x)) &\to 2nd(x' : (x : inf(s(x))))\\
inf(x' : inf(x)) &\to inf(x' : (x : inf(s(x))))
\end{aligned}
$$

Note that here $\mathbb{T}(\mathcal{R},\Pi)$ is different from the transformed system in Example 44, since there we used a typed version of $\mathcal{R}$.

This loss of structure regarding the partition of function symbols into defined symbols and constructors but also the fact that all function symbols occur in right-hand sides of some rules makes a dependency pair analysis much harder in practice. Several processors and methods (like dependency graph processors, narrowing processors or processors relying on usable rules) are not applicable or yield weaker results in the analysis of TRSs obtained by $\mathbb{T}$. This is particularly bad for dependency graph processors, because these processors are arguably among the most important processors used in typical dependency pair proofs of termination. Hence, in this subsection we propose a slight optimization of this processor in order to improve its performance for TRSs obtained by $\mathbb{T}$.

Roughly, a dependency graph processor when given a DP problem $(\mathcal{P}, \mathcal{R})$ tries to construct the dependency graph whose nodes are the dependency pairs from $\mathcal{R}$ with edges from $s_1 \to t_1$ to $s_2 \to t_2$ if there exist substitutions σ_1, σ_2 such that

$$t_1\sigma_1 \to^*_{\mathcal{R}} s_2\sigma_2. \tag{4.8}$$

Then, the dependency graph is decomposed into strongly connected components (SCC) whose nodes form the subproblems the processor returns.

The major obstacle when applying this processor is the fact that (4.8) is undecidable. Hence, in practice one uses (over-)approximations. Typically, these approximations rely on identifying a “constant” part of the term t_1, i.e. a part of t_1 that cannot be altered by reduction of $t_1\sigma_1$. This is to say that $t_1 = C[t_1^1, \ldots, t_1^n]_{p_1,\ldots,p_n}$ and for every term $\bar{t}$ with $t_1\sigma_1 \to^*_{\mathcal{R}} \bar{t}$ we have $\bar{t} = C[\bar{t}_1^1, \ldots, \bar{t}_1^n]_{p_1,\ldots,p_n}\tau$ (for some substitution τ).

In most classical approaches to such approximations the “constant” part of t that is identified is not only constant but irreducible, i.e. for every term $\bar{t}$ with $t_1\sigma_1 \to^*_{\mathcal{R}} \bar{t}$ every reduction step in this reduction sequence occurs at or below p_i for some $1 \leq i \leq n$. However, for TRSs obtained by $\mathbb{T}$ this kind of approximation is not feasible, since we have to consider many rules of the shape $C[l] \to C[r]$ for some context C. When such a rule is applied to a term (say at the root position) there might still be a constant part of the term (namely C) even though the term is reducible.

To exploit this observation we need to take a closer look at the rules of TRSs obtained by $\mathbb{T}$. In particular, we are interested in function symbols f that are constructors in some TRS $\mathcal{R}$ with forbidden patterns Π but defined symbols in $\mathbb{T}(\mathcal{R}, \Pi)$. According to Definition 28, the rules of $\mathbb{T}(\mathcal{R}, \Pi)$, defining function symbols f that were constructors in $\mathcal{R}$, are of the shape

$$f(l_1, \ldots, l_n) \to f(r_1, \ldots, r_n)$$

Hence, these function symbols behave like constructors in that they cannot be altered by reduction steps occurring at, parallel or below their occurrences in the term. This motivates the definition of *quasi-defined symbols* which are function symbols that might change through reduction.

Definition 29 (quasi-defined symbols). *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS. For each rule $l \to r$ (where $l \neq r$) the maximal context C is identified, such that $l = C[s_1 \dots s_n]$ and $r = C[t_1 \dots t_n]$ (note that C might be empty) and $root(s_i) \in \mathcal{F}$ for all $1 \leq i \leq n$. Then $root(s_i)$ is a* quasi-defined symbol *for all $1 \leq i \leq n$.*

We denote the set of quasi-defined symbols of a TRS $(R, \mathcal{F})$ by D^q and the corresponding set of quasi-constructors (i.e., $\mathcal{F} \setminus D^q$) as C^q. Observe that quasi-defined function symbols and quasi-constructors are incomparable with the usual notions of defined functions and constructors.

Example 47. *Consider the TRS $\mathcal{R}$ given by*

$$g(f(x)) \to g(f(a))$$

Here, f is quasi-defined but not defined and g is defined but not quasi-defined.

Like ordinary constructors, quasi-constructors have the important property that the "top" part of a term - if it is constructed only from such constructors - cannot be modified by reduction, i.e., whenever $s = C[s_1, \dots, s_n]$ and C consists only of functions from C^q and variables, then for all terms t with $s \to^* t$ we have $t = C[t_1, \dots, t_n]$ for some terms $t_1, \dots, t_n$.

This motivates the definition of the function $\widetilde{cap}$ that replaces all maximal subterms rooted by functions from D^q by fresh variables. One can use this function to approximate the existence of an edge from the DP $s_1 \to t_1$ to $s_2 \to t_2$ by unifying $ren(\widetilde{cap}(t_1))$ with s_2. If $ren(\widetilde{cap}(t_1))$ is unifiable with s_2, then the edge is drawn, otherwise there is no edge. Here, the function ren replaces each variable in the given term by a fresh variable. Thus, ren maps terms to linear terms (cf. e.g. [33] for a similar approximation of dependency graphs using the classical notions of defined function symbols and constructors).

Lemma 17. *Let $(\mathcal{P}, \mathcal{R})$ be a DP problem and $s_1 \to t_1, s_2 \to t_2 \in \mathcal{P}$ be two dependency pairs. If there exist substitutions σ_1, σ_2 such that $t_1\sigma_1 \to_{\mathcal{R}}^* s_2\sigma_2$, then $ren(\widetilde{cap}(t_1))$ unifies with s_2 (after renaming the variables to make the terms variable disjoint).*

Proof. Assume that $ren(\widetilde{cap}(t_1))$ and s_2 do not unify. Then, consider some term $ren(\widetilde{cap}(t_1))\sigma$ and a reduction step $ren(\widetilde{cap}(t_1))\sigma \xrightarrow{p}_{\mathcal{R}} t$. We are going to prove that $t = ren(\widetilde{cap}(t_1))\sigma'$ for some substitution σ'. First, assume $p \notin Pos_{\mathcal{F}}(ren(\widetilde{cap}(t_1)))$. Then $t = ren(\widetilde{cap}(t_1))\sigma'$ follows immediately, since $ren(\widetilde{cap}(t_1))$ is linear.

On the other hand, if $p \in Pos_{\mathcal{F}}(ren(\widetilde{cap}(t_1)))$, then the applied rewrite rule must be of the shape $C[l_1, \dots, l_n]_{p_1,\dots,p_n} \to C[r_1, \dots, r_n]_{p_1,\dots,p_n}$ where for each p_i ($1 \leq i \leq n$) $p.p_i \geq q$ for some $q \in Pos_V(ren(\widetilde{cap}(t_1)))$, because otherwise $ren(\widetilde{cap}(t_1))$ would contain a quasi-defined symbol according to Definition 29 which it does not by the definition of $\widetilde{cap}$. Hence, because of linearity of $ren(\widetilde{cap}(t_1))$ we get $t = ren(\widetilde{cap}(t_1))\sigma'$ for some σ'.

An easy induction yields that $\bar{t} = ren(\widetilde{cap}(t_1))\sigma'$ for some σ' for every term $\bar{t}$ with $ren(\widetilde{cap}(t_1))\sigma \xrightarrow{p}{}^{*}_{\mathcal{R}} \bar{t}$.

Now assume - contradicting our assumption that $ren(\widetilde{cap}(t_1))$ and s_2 do not unify - that there exist substitutions σ_1 and σ_2 such that $t_1\sigma_1 \to^{*}_{\mathcal{R}} s_2\sigma_2$. We already proved that this implies $s_2\sigma_2 = ren(\widetilde{cap}(t_1))\sigma'$ for some σ', which further implies that s_2 and $ren(\widetilde{cap}(t_1))$ are unifiable after renaming their variables. Hence, we get a contradiction to our assumption of non-unifiability of these terms. □

Example 48. *Consider the TRS $\mathcal{R}$ from Example 46. There, $: \in C^q$ while $:$ is not a constructor (in the traditional sense). Thus, consider for instance a dependency pair $inf(x) :^{\#} y \to (x : inf(s(x))) :^{\#} y \in DP(\mathcal{R})$ (i.e. $C[inf(x)] \to C[x : inf(s(x))]$)). Then $\widetilde{cap}^{\mu}(ren^{\mu}((x : inf(s(x))) :^{\#} y)) = (x : z) :^{\#} y$ which is not unifiable with $inf(x) : y$ hence there is no arc in the estimated dependency graph from this dependency pair to itself. Indeed, the termination tool VMTL (cf. Chapter 5 below) that contains an implementation of the dependency graph approximation based on $\widetilde{cap}$ can automatically prove termination of $\mathcal{R}$ but fails to do so if the use of $\widetilde{cap}$ is disabled.*

4.4.2 A Modified Dependency Pair Framework for Rewriting with Forbidden Patterns

In this section we propose a modification of the well-known dependency pair (DP) framework of [33] (which is in turn based on dependency pairs of [9]) that incorporates forbidden pattern restrictions. Compared to the transformation based approach of Section 4.4.1.1 this approach works for a more general class of forbidden patterns, namely forbidden h- and b-patterns. Hence, in this section we are exclusively concerned with these kinds of patterns.

The central observation of the (ordinary) dependency pair approach is that given a non-terminating rewrite system $\mathcal{R}$, there exists an infinite reduction sequence (starting w.l.o.g. with a root reduction step), such that no redex contracted in this sequence contains a non-terminating proper subterm. Such reduction sequences roughly correspond to minimal dependency pair chains whose existence or non-existence is analyzed in the DP framework. For rewriting with forbidden patterns the above observation does not hold.

Example 49. *Consider the following TRS $\mathcal{R}$*

$$a \to f(a) \qquad\qquad f(x) \to g(x)$$

and an associated set of forbidden patterns $\Pi = \{\langle f(x), 1, h\rangle\}$. $\mathcal{R}$ is not Π-terminating: $a \to_{\Pi} f(a) \to_{\Pi} g(a) \to_{\Pi} g(f(a)) \to_{\Pi} g(g(a)) \to_{\Pi} \dots$ Note that since position 1 is forbidden in $f(a)$, we do not have $f(a) \to_{\Pi} f(f(a))$. Obviously, every non-Π-terminating term s must contain exactly one a. After this a is reduced, the single a-symbol in the contracted term is forbidden (as it occurs in the first argument of f). Hence, the redex of the following reduction must properly contain a.

In Example 49 reductions whose redexes properly contain non-Π-terminating terms are crucial for the existence of infinite Π-derivations. Hence, instead of ordinary non-termination we focus on a restricted form of non-Π-termination, namely non-Π-termination in a context.

Definition 30 (termination in a context)**.** *Let $\mathcal{R}$ be a TRS and* Π *be a set of forbidden patterns. A term s is* Π-terminating in context $C[\Box]_p$ *if $C[s]_p$ does not admit an infinite* Π*-reduction sequence where each redex contracted occurs at, below or parallel to p and where infinitely many steps are at or below p.*

We omit explicit reference to the context if it is clear which one is meant. For instance the term $s|_p$ is Π-terminating in its context means that $s|_p$ is Π-terminating in the context $s[\Box]_p$.

We say a term s is *minimal* non-Π-terminating in a context $C[\Box]_q$ (w.r.t. a rewrite system $\mathcal{R}$ and a set of forbidden patterns Π) if s is non-Π-terminating in $C[\Box]_q$ and every proper subterm $s|_p$ of s is Π-terminating in $C[s[\Box]_p]_q$. The following lemma provides some insight into the shape of infinite Π-reduction sequences starting from minimal non-terminating terms.

Lemma 18. *Let $\mathcal{R}$ be a TRS and let* Π *be a set of forbidden h- and b-patterns. A term s that is minimal non-*Π*-terminating in a context $C[\Box]_q$ admits a reduction sequence*

$$C[s]_q \xrightarrow{\not\leq q}{}^{*}_{\Pi} C'[s']_q = C'[l\sigma]_q \xrightarrow{q}_{\Pi} C'[r\sigma]_q = C'[t]_q$$

*such that t contains a subterm $t|_p$ that is minimal non-*Π*-terminating in the context $C'[t[\Box]_p]_q$.*

Proof. As s is not Π-terminating in $C[\Box]_q$, there is an infinite Π-reduction sequence starting from $C[s]_q$ such that all reduction steps are at, below or parallel to q and infinitely many reduction steps are at or below q (according to Definition 30). Since, s is minimally non-Π-terminating in $C[\Box]_q$, eventually there must be a step at position q in this reduction sequence (as otherwise infinitely many reduction steps occurred at or below a proper subterm of s contradicting termination of these subterms in their contexts). Hence, we have

$$C[s]_q \xrightarrow{\not\leq q}{}^{*}_{\Pi} C'[s']_q = C'[l\sigma]_q \xrightarrow{q}_{\Pi} C'[r\sigma]_q = C'[t]_q$$

as part of our infinite Π-reduction sequence. Because of infinity of the reduction sequence t must be non-Π-terminating in $C'[\Box]_q$. However, every term t that is non-Π-terminating in a context $C'[\Box]_q$ has a subterm $t|_p$ that is minimally non-Π-terminating in $C'[t[\Box]_p]_q$. □

Note that in contrast to ordinary rewriting and standard minimal non-terminating terms one can in general not assume that $p \in Pos_{\mathcal{F}}(r)$ (this effect similarly exists in context-sensitive rewriting, cf. [4, 2]).

Example 50. *Consider $\mathcal{R}$ and Π of Example 49 and the term $f(a)$ which is minimal non-Π-terminating (in the empty context), since position 1 is forbidden in $f(a)$ according to Π. Now consider the reduction $f(a) = f(x)\sigma \overset{\epsilon}{\to}_{\Pi} g(a) = g(x)\sigma$ ($x\sigma = a$). The term $g(a)$ contains only one proper minimal non-Π-terminating subterm $g(a)|_1 = a$ despite the fact that $1 \notin Pos_{\mathcal{F}}(g(x))$.*

In our approach we pay attention to this phenomenon by having additional dependency pairs to explicitly mimic the necessary extractions of (minimal non-Π-terminating) subterms in DP chains (cf. V_c, A_c and S_c in Definition 32 below). Technically, these rules (which we call structural dependency pairs) model the explicit extraction of minimal non-Π-terminating terms in DP-chains and the introduction of the suitable dependency pair symbol at the root of these terms. This mechanism is similar to the way migrating variables are dealt with in the context-sensitive dependency pair approach of [2]. However, there using the concepts of "hidden terms" and "function symbols hiding positions" it is sufficient to perform subterm extractions out of contexts of hidden terms in right-hand sides of rewrite rules and over arguments of functions hidden by the function.

In the case of forbidden patterns it is necessary to use a more general mechanism of subterm extraction, since whether a term is hidden within the right-hand side of a rewrite rule (i.e., forbidden but might eventually be activated) may depend on the context the right-hand side of the rule is located in and the concrete instance of this right-hand side. Hence, in sharp contrast to the context-sensitive dependency pair framework of [2] the structural forbidden pattern dependency pairs associated to a TRS model subterm extractions out of arbitrary contexts (cf. V_c, A_c and S_c in Definition 32 below).

However, we cannot disregard the contexts from which minimal non-Π-terminating terms are extracted in DP-chains, since these contexts may contribute to the matching of a forbidden pattern thus influencing the status of some position in the minimal non-Π-terminating term. In order to keep track of the subterm extractions in dependency pair chains a context is associated to each dependency pair. It represents the context from which a minimal non-Π-terminating term is extracted when the dependency pair is applied.

Informally, this amounts to an extended *contextual* version of dependency pairs which incorporates the full information of the given rules (especially the complete right-hand sides) in the form of associated contexts, but which still enables the typical DP-based reasoning enriched by *structural* DP-rules that can descend into variable subterms of right-hand sides as well as to control where subsequent DP-reductions are allowed to take place.

Before defining contextual dependency pairs we observe that sometimes positions of right-hand sides are forbidden regardless of the instantiation or location (in a context) of this right-hand side. In particular, positions forbidden by *stable* forbidden patterns have this property. We will use this observation to reduce the number of dependency pairs that we have to consider (cf. Definition 32 below).

Definition 31 (stable forbidden pattern)**.** *Given a rewrite system $\mathcal{R}$, a forbidden pattern $\pi = (t, p, \lambda)$ is called* stable *if t is linear and no rule overlaps t at any (function symbol) position parallel to p if $\lambda = b$ and no rule overlaps t at any (function symbol) position parallel to or below p if $\lambda = h$. By $Stb(\Pi)$ we denote the subset of stable forbidden patterns of Π.*

Note that Definition 31 is in a certain way dual to Item 1 of Definition 19 and Definition 20 where forbidden patterns were not allowed to be overlapped by right-hand sides of rules parallel to the forbidden position in case of h-patterns not overlapped at all by right-hand sides of rules for b-patterns. This duality is reflected in the effects the respective restrictions on forbidden patterns have on the restricted rewrite relation. In the case of Definitions 19 and 20 this effect was that the status of allowed positions does not change during reductions parallel to these positions (Proposition 6). Dually, in the case of stable forbidden patterns the status of forbidden positions does not change during reductions parallel to or below these positions.

Lemma 19. *Let $\mathcal{R}$ be a rewrite system and let π be a stable pattern matching a subterm $s|_p$ of a term s (and thus forbidding some position $p.q$ in s). If $s \xrightarrow{p'}_{\mathcal{R},\Pi} t$ for some $p' \parallel p.q$ or $p' > p.q$, then $p.q$ is also forbidden in t.*

Proof. Let $\pi = \langle u, o, \lambda \rangle$. First, if $\lambda = b$, then p' cannot be below $p.q$, since positions below $p.q$ are forbidden by π. Hence, p' is parallel to $p.q$. However, since u is linear and not overlapped by any rewrite rule from $\mathcal{R}$ parallel to o, u matches $t|_p$ and thus $p.q$ is also forbidden in t.

Second, if $\lambda = h$, then u is not overlapped by any rule of $\mathcal{R}$ parallel to o or below o. Hence, (also because u is linear) u matches $t|_p$ and thus $p.q$ is forbidden in t. □

We are now ready to define the notion of contextual dependency pairs (CDPs) associated to a rewrite system with forbidden patterns, CDP-problems and CDP-chains.

Definition 32 (extended *contextual* dependency pairs)**.** *Let $(\mathcal{F}, R)$ be a TRS where the signature is partitioned into defined symbols $\mathcal{D}$ and constructors $\mathcal{C}$. The set of (extended)* contextual dependency pairs (CDPs) *$CDP(\mathcal{R})$ is given by $DP_c(\mathcal{R}) \uplus V_c(\mathcal{R}) \uplus A_c(\mathcal{R}) \uplus S_c(\mathcal{R})$, where*

$$\begin{array}{rcl}
DP_c(\mathcal{R}) & = & \{l^\# \to r|_p^\# \; [c] \mid l \to r \in R, p \in Pos_{\mathcal{D}}^{Stb(\Pi)}(r), c = r[\Box]_p\} \\
V_c(\mathcal{R}) & = & \{l^\# \to T(r|_p) \; [c] \mid l \to r \in R, r|_p = x \in Var, c = r[\Box]_p\} \\
A_c(\mathcal{R}) & = & \{T(f(x_1, \ldots, x_{ar(f)})) \to f^\#(x_1, \ldots, x_{ar(f)}) \; [\Box] \mid l \to r \in R, \\
 & & \hfill root(r|_p) = f \in \mathcal{D}\} \\
S_c(\mathcal{R}) & = & \{T(f(\vec{x})) \to T(x_i)[f(\vec{x})[\Box]_i] \mid \vec{x} = x_1, \ldots, x_{ar(f)}, l \to r \in R, \\
 & & \hfill root(r|_p) = f, i \in \{1, \ldots, ar(f)\}\}\,.
\end{array}$$

Here, T is a new auxiliary function symbol (the token *symbol for "shifting attention"). We call $V_c(\mathcal{R})$* variable descent *CDPs, $S_c(\mathcal{R})$* shift *CDPs and $A_c(\mathcal{R})$* activation *CDPs.*

Example 51. *Consider the TRS $\mathcal{R}$ of Example 49. Here, $CDP(\mathcal{R})$ consists of:*

$$\begin{array}{rclrclrcl}
a^\# & \to & a^\#[f(\Box)] & a^\# & \to & f^\#(a)[\Box] & f^\#(x) & \to & T(x)[g(\Box)] \\
T(a) & \to & a^\#[\Box] & T(f(x)) & \to & f^\#(x)[\Box] & T(g(x)) & \to & g^\#(x)[\Box] \\
T(f(x)) & \to & T(x)[f(\Box)] & T(g(x)) & \to & T(x)[g(\Box)] & & &
\end{array}$$

Contextual rules of the shape $l \to r\ [c]$ can be interpreted as $l \to c[r]$ (provided that $Var(c[r]) \subseteq Var(l)$) when used as rewrite rules. Slightly abusing notation, for a set $\mathcal{P}$ of such contextual rewrite rules (i.e. a *contextual TRS*) we denote by $\to_{\mathcal{P}}$ the corresponding induced ordinary rewrite relation.

Definition 33 (forbidden pattern CDP problem)**.** *A forbidden pattern CDP problem (FP-CDP problem or just CDP problem) is a quadruple $(\mathcal{P}, \mathcal{R}, \Pi, T)$ where $\mathcal{P}$ is a contextual TRS, $\mathcal{R} = (\mathcal{F}, R)$ is a TRS, Π is a set of forbidden patterns over $\mathcal{F}$ and T is a designated function symbol with $T \notin \mathcal{F}$ that occurs only at the root position of left- and right-hand sides of rules in $\mathcal{P}$ (but not e.g. in contexts).*

Definition 34 (forbidden pattern CDP chain)**.** *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be a CDP problem where $\mathcal{R} = (\mathcal{F}, R)$. The sequence $S\colon s_1 \to t_1\ [c_1[\Box]_{p_1}], s_2 \to t_2\ [c_2[\Box]_{p_2}], \ldots$ is a $(\mathcal{P}, \mathcal{R}, \Pi, T)$-CDP chain (we also say FP-CDP chain or just CDP chain if the CDP problem is clear from the context) if*

- *there exists a substitution $\sigma\colon Var \to \mathcal{T}(\mathcal{F}, V)$, such that*

$$\begin{array}{rcl}
s_1\sigma & \to_{\mathcal{P}} & c_1[t_1\sigma]_{p_1} = c'_1[t_1\sigma]_{p'_1} \\
\xrightarrow{\not\leq p'_1 *}_{\mathcal{R}} c''_1[s_2\sigma]_{p'_1} & \to_{\mathcal{P}} & c''_1[c_2[t_2\sigma]_{p_2}]_{p'_1} = c'_2[t_2\sigma]_{p'_2} \\
\xrightarrow{\not\leq p'_2 *}_{\mathcal{R}} c''_2[s_3\sigma]_{p'_2} & \to_{\mathcal{P}} & c''_2[c_3[t_3\sigma]_{p_3}]_{p'_2} = c'_3[t_3\sigma]_{p'_3} \ \ldots
\end{array}$$

  where $c'_i = c''_{i-1}[c_i]$ and $p'_i = p'_{i-1}p_i$ for all $1 \leq i$ ($s_1\sigma = c''_0[s_1\sigma]_{p'_0}$ with $p'_0 = \epsilon$, $c''_0 = \Box$),

- *the $\mathcal{R}$-reduction $c'_i[t_i\sigma]_{p'_i} \xrightarrow{\not\leq p'_i *}_{\mathcal{R}} c''_i[s_{i+1}\sigma]_{p'_i}$ is empty (i.e., $c'_i[t_i\sigma]_{p'_i} = c''_i[s_{i+1}\sigma]_{p'_i}$) whenever $root(t_i) = T$ (i.e., the token symbol), and*

- *for each single reduction $s \xrightarrow{q}_{\mathcal{P}} t$ or $s \xrightarrow{q}_{\mathcal{R}} t$ in this reduction sequence position q is allowed in $erase(s)$ according to Π. Here $erase(s)$ is obtained from s by replacing all marked dependency pair symbols $f^\#$ by their unmarked versions f and by replacing terms $T(s')$ by s'.*[3]

Moreover, S is minimal if for every $i \geq 0$ every subterm of $c'_i[t_i\sigma]_{p'_i}$ at position $q > p'_i$ is Π-terminating in its context (w.r.t. $\mathcal{R}$).

[3]Note that this definition makes sense since whenever a T occurs in s, then q is not below the occurrence of T. Moreover, this definition of *erase* is formally not compatible with the DP framework, since it is based on the correspondence of marked dependency pair symbols and original function symbols they originated from. This correspondence might not exist in arbitrary CDP problems. However, to restore full modularity the *erase* function could be made part of the notion of CDP problem. We refrain from doing so for notational simplicity.

Example 52. *Consider the TRS $\mathcal{R}$ and Π from Example 49 ($CDP(\mathcal{R})$ is given in Example 51) and the corresponding FP-CDP $P = (CDP(\mathcal{R}), \mathcal{R}, \Pi, T)$. P admits an infinite CDP chain:*

$$a^\# \to f^\#(a)\ [\Box], f^\#(x) \to T(x)\ [g(\Box)], T(a) \to a^\#\ [\Box], \ldots$$

corresponding to

$$a^\# \to_{DP_c(\mathcal{R})} f^\#(a) \to_{V_c(\mathcal{R})} g(T(a)) \to_{A_c(\mathcal{R})} g(a^\#) \to_{DP_c(\mathcal{R})} g(f^\#(a)) \ldots$$

We say a CDP problem is *finite* if it does not admit an infinite minimal CDP chain. Indeed, the existence of infinite $(CDP(\mathcal{R}), \mathcal{R}, \Pi, T)$-chains coincides with non-Π-termination of $\mathcal{R}$. Before proving this we provide a lemma stating that forbidden rewrite steps can be extracted out of contexts.

Lemma 20 (extraction lemma)**.** *If $C[s]_p \xrightarrow{\geq p}_\Pi C[t]_p$, then $s \to_\Pi t$.*

Proof. Immediate by the definition of rewriting with forbidden patterns. □

Theorem 15. *Let $\mathcal{R}$ be a TRS with an associated set of forbidden h- and b-patterns Π. $\mathcal{R}$ is Π-terminating if and only if the FP-CDP problem $(CDP(\mathcal{R}), \mathcal{R}, \Pi, T)$ is finite.*

Proof. IF: Let $\mathcal{R}$ be non-Π-terminating. According to Lemma 18, there exist terms s, s_i, t_i and t'_i such that

$$\begin{aligned} S : s &\xrightarrow{>\epsilon}{}^*_\Pi\ t_1 \xrightarrow{\epsilon}_\Pi s_1 = C'_2[t'_2]_{p_2} \\ &\xrightarrow{\not\leq p_2}{}^*_\Pi\ C_2[t_2]_{p_2} \xrightarrow{p_2}_\Pi C_2[s_2]_{p_2} = C'_3[t'_3]_{p_3} \\ &\xrightarrow{\not\leq p_3}{}^*_\Pi\ C_3[t_3]_{p_3} \xrightarrow{p_3}_\Pi C_3[s_3]_{p_3} = C'_4[t'_4]_{p_4} \ldots \end{aligned}$$

$p_i \leq p_{i+1}$, t'_i is minimally non-Π-terminating in $C'_i[\Box]_{p_i}$ for all $i \geq 1$, and every proper subterm of s is Π-terminating (regardless of the context, hence s is in particular minimal non-Π-terminating in $\Box$). Here, $p_1 = \epsilon$, $C_1 = C'_1 = Box$ and $t'_1 = t_1$.

We are going to construct an infinite $(CDP(\mathcal{R}), \mathcal{R}, \Pi, T)$-chain T by associating a (sequence of) CDPs to each $C_i[t_i]_{p_i} \xrightarrow{p_i}_\Pi C_i[s_i]_{p_i}$ step. Consider one of these reduction steps $C_i[t_i]_{p_i} \xrightarrow{p_i}_{\Pi, l \to r} C_i[s_i]_{p_i} = C'_{i+1}[t'_{i+1}]_{p_{i+1}}$. Let $p_i.q = p_{i+1}$; we distinguish 2 cases:

First, if $q \in Pos_\mathcal{F}(r)$, then the CDP $l^\# \to r|^\#_q[c] \in DP_c(\mathcal{R})$ is used. Note that $root(r|_q) \in \mathcal{D}$, as t'_{i+1} is minimally non-Π-terminating. Moreover, $q \in Pos^{Stb(\Pi)}$, since otherwise p_{i+1} would be forbidden by a stable forbidden pattern in $C'_{i+1}[t'_{i+1}]_{p_{i+1}}$ and thus also in every term obtained from $C'_{i+1}[t'_{i+1}]_{p_{i+1}}$ through reduction parallel to or below p_{i+1}, due to Lemma 19. Hence, there could not be a further step at position p_{i+1} contradicting the existence of a reduction chain of the above shape. Finally, we also have $C_i[c] = C'_{i+1}$ by Definition 32.

Second, if $q \notin Pos_\mathcal{F}(r)$, let $q' \leq q$ be the unique variable position of r that is above q. Now we construct a sequence of CDPs starting with $l^\# \to T(x)[c] \in V(\mathcal{R})$

where $c = r[\Box]_{q'}$. By using this CDP we "introduce" the token symbol T at position q' in s_i. The goal now is to shift it to position q.

In the following we say that a function symbol f is a *shift symbol* if there exist CDPs $T(f(\vec{x})) \to T(x_i)[f(\vec{x})[\Box]_i]$ for all $i \in \{1, \ldots, ar(f)\}$. Assume $q' \neq q$ (say $q'.i.o = q$) and let $root(s_i|_{q'}) = f$.

If f is not a shift symbol, then f does not occur in the right-hand side of a rewrite rule at all (according to Definition 32). However, if f does not occur in the right-hand side of any rule of $\mathcal{R}$, $s_i|_{q'}$ must be the descendant of some proper subterm of s. However, $s_i|_{q'}$ is non-Π-terminating, since it contains t'_{i+1} which is non-Π-terminating in its context. Thus $s_i|_{q'}$ cannot be a successor of such a proper subterm of s, since these subterms were assumed to be Π-terminating (in any context) (cf. also Lemma 20).

Hence, f is a shift symbol and thus there is a CDP $T(f(x_1, \ldots, x_{ar(f)})) \to T(x_i)[c] \in S_c(\mathcal{R})$ where $f(x_1, \ldots, x_{i-1}, \Box, x_{i+1}, \ldots, x_{ar(f)}) = c$. By adding this CDP we shift the token symbol to position $q'.i$ in s_i (more precisely with the addition of the shift CDP we are now considering a term s'_i with $erase(s'_i) = erase(s_i)$ where the unique occurrence of the token symbol is at position $q'.i$). If $q'.i \neq q$ we add more CDPs from $S_c(\mathcal{R})$ to shift the token symbol to $q'.i.i'$, $q'.i.i'.i''$, …, until the token is finally shifted to q.

Finally, we add the activation CDP $T(g(\vec{x})) \to g^{\#}(\vec{x})[\Box] \in A_c(\mathcal{R})$, where $g = root(s_i|_q)$. Note that, as for the shift CDPs, here g must occur in the right-hand side of some rewrite rule, since otherwise $s_i|_q$ would be the descendent of some proper subterm of s which contradicts non-Π-termination of t'_{i+1}.

Moreover, since $s_i|_q$ is minimal non-Π-terminating, we have $root(s_i|_q) \in \mathcal{D}$.

It is easy to see that the infinite sequence T of CDPs obtained by this construction actually forms an infinite CDP chain, where σ is given by the substitutions used in the $C_i[t_i]_{p_i} \xrightarrow{p_i}_{\Pi} C_i[s_i]_{p_i}$ steps of S (note that we consider CDPs in chains to be variable disjoint). The fact that we actually have a valid CDP chain is a direct consequence of the particular choice of S.

ONLY IF: If there exists an infinite CDP-chain we obtain an infinite $\mathcal{R}$-reduction by considering the $(CDP(\mathcal{R}) \cup \mathcal{R})$-reduction of Definition 34. Then by applying *erase* to every term in this chain, we get that every single $(CDP(\mathcal{R}) \cup \mathcal{R})$-step can be simulated by 0 or 1 $\to_{\mathcal{R}}$-reduction steps. Here the simulating reduction is empty only if a $CDP(\mathcal{R})$-step with rules from $S_c(\mathcal{R})$ or $A_c(\mathcal{R})$ occurs. However, it is easy to see that no infinite $(CDP(\mathcal{R}) \cup \mathcal{R})$-reduction sequence can use only these rules, hence the simulating $\mathcal{R}$-reduction is infinite as well. $\Box$

Now, following the dependency pair framework of [33] we define CDP processors as functions mapping CDP problems to sets of CDP problems.

It is easy to observe that each FP-CDP chain w.r.t. some FP-CDP problem $(\mathcal{P}, \mathcal{R}, \Pi, T)$ is also an ordinary (though not necessarily minimal) DP chain w.r.t. $(\mathcal{P}, \mathcal{R})$ (when disregarding the contexts of DPs). Hence, in some cases processors that are sound in the ordinary DP framework of [33] and do not rely on minimality can be adapted to work also in the forbidden pattern contextual extension of the

DP framework. One example of such a processor is the reduction pair processors not using usable rules ([33]). Another important example is the dependency graph processor. Both processors have been used in our experiments. In both cases, given a CDP problem $(\mathcal{P}, \mathcal{R}, \Pi, T)$, the processors are applied to the ordinary DP problem $(\mathcal{P}', \mathcal{R})$, where $\mathcal{P}'$ is obtained from $\mathcal{P}$ by stripping the contextual rules of their contexts.

CDP Processors

In the following we develop a method to prove the absence of minimal CDP chains by inspecting the contexts of dependency pairs. To this end we consider the nested contexts of consecutive dependency pairs of candidates for infinite CDP chains. Then, if for such a candidate in the obtained nested contexts of consecutive dependency pairs the unique box position is forbidden (by certain forbidden patterns), the candidate chain is not a proper FP-CDP chain. A CDP processor can then soundly delete a CDP $s \to t[c]$ from a CDP-problem if no candidate chain containing $s \to t[c]$ is a proper FP-CDP chain (provided that the set of candidates is complete).

Example 53. *Consider a CDP problem $(\mathcal{P}, \mathcal{R}, \Pi, T)$ where*

$$\begin{array}{rclcrcl} \mathcal{P} & = & \{a^\# \to a^\#[f(\Box)]\} & & \mathcal{R} & = & \{a \to f(a)\} \\ \Pi & = & \{\langle f(f(f(x))), 1.1, b\rangle\}. & & & & \end{array}$$

If there were an infinite FP-CDP chain w.r.t. this CDP problem, then it would consist of an infinite sequence of the only CDP $a^\# \to a^\#[f(\Box)]$. Hence, this sequence is the only candidate for an FP-CDP chain. Now considering the contexts occurring in this CDP chain candidate we get $f(f(...(\Box)...))$ (for any sufficiently large finite subsequence). However, in this term the box position is forbidden by Π. Hence, the CDP chain candidate is not a proper FP-CDP chain and since it was the only candidate, we conclude finiteness of the CDP problem.

Unfortunately, there are two major problems with this approach. First, in order to obtain a sound CDP processor one would have to consider candidates for CDP chains in a complete way. Second, according to Definition 34 contexts are not constant but may be modified at positions parallel to the box position in FP-CDP chains.

We will deal with the second problem first, starting with the observation that the (nested) contexts are stable modulo reductions parallel to the position of the hole, i.e. they are altered only through reductions parallel to the hole position. Hence, if forbidden patterns oblivious to this kind of parallel reductions forbid the hole position in such a context, the corresponding sequence of dependency pairs cannot form an FP-CDP chain according to Definition 34. We characterize (or rather approximate) these patterns by the definition of the subset Π_{orth} of Π. The name Π_{orth} expresses that these forbidden patterns are orthogonal to a given rewrite system $\mathcal{R}$ in that they are not overlapped by rules of $\mathcal{R}$.

Definition 35 (Π_{orth})**.** *Let $\mathcal{R}$ be a TRS and Π be a set of corresponding forbidden patterns. The set $\Pi_{orth} \subseteq \Pi$ consists of those forbidden patterns $\langle t, p, \lambda\rangle$ where $\lambda \in \{h, b\}$, t is linear and not overlapped by any rule of $\mathcal{R}$ at any (function symbol) position that is parallel to or below p.*

Note that there is a subtle difference between the definition of stable patterns (Definition 31) and that of patterns orthogonal to $\mathcal{R}$. In the definition of Π_{orth}, in contrast to that of $Stb(\Pi)$, also overlaps at forbidden positions are ruled out. So we have $\Pi_{orth} \subseteq Stb(\Pi) \subseteq \Pi$ for any given set Π of forbidden patterns and rewrite system $\mathcal{R}$. The reason for this more restricted definition of Π_{orth} becomes obvious in Lemma 21 below, where the terms in question can be arbitrarily reduced with $\to_{\mathcal{R}}$ (i.e. reductions not adhering to the forbidden pattern restrictions) while matches by forbidden patterns are preserved. Indeed, the lemma would not hold if we used $Stb(\Pi)$ instead of Π_{orth}.

Lemma 21 is the key result for analyzing nested contexts of CDP chain candidates. It states that whenever the box position q of a nested context corresponding to a CDP chain candidate (after substituting the right-hand side of the last CDP) is forbidden, then this position is also forbidden in every other term obtained from the nested context by rewriting at positions parallel to q.

Lemma 21. *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be an FP-CDP problem and let $s_1 \to t_1[c_1], \dots, s_n \to t_n[c_n]$ be a sequence of CDPs. If position $p_1. \cdots .p_n$ is forbidden in the term*

$$c_1[c_2[\dots c_n[erase(t_n)]_{p_n} \dots]_{p_2}]_{p_1}$$

by a forbidden pattern from Π_{orth}, then the same position is forbidden in

$$c'_1[c'_2[\dots c'_n[t'_n]_{p_n} \dots]_{p_2}]_{p_1}$$

where $c_i \to^*_{\mathcal{R}} c'_i$ with reductions parallel to p_i for all $1 \leq i \leq n$ and $erase(t_n) \overset{>\epsilon}{\to}{}^*_{\mathcal{R}} t'_n$.

Proof. For brevity let

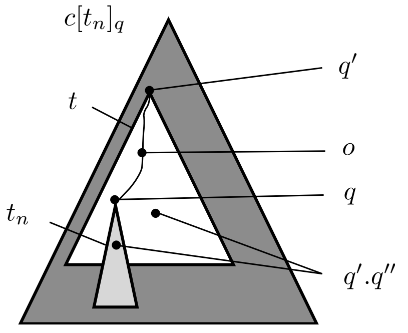
$$c[t_n]_q = c_1[c_2[\dots c_n[erase(t_n)]_{p_n} \dots]_{p_2}]_{p_1},$$

where $q = p_1.p_2. \cdots .p_n$ and let

$$c'[t'_n]_q \text{ be some } c'_1[c'_2[\dots c'_n[t'_n]_{p_n} \dots]_{p_2}]_{p_1}$$

where $c_i \to^*_{\mathcal{R}} c'_i$ with reductions parallel to p_i for all $1 \leq i \leq n$ and $erase(t_n) \overset{>\epsilon}{\to}{}^*_{\mathcal{R}} t'_n$.

Assume the forbidden pattern $\langle t, o, \lambda\rangle$ forbidding the reduction of $c[t_n]_q$ at position q matches the term at position $q' < q$ and assume moreover that the same pattern does not match $c'[t_n]_q$. Since we consider plain $\mathcal{R}$-reduction and not forbidden pattern reduction, we have $S\colon c[t_n]_q \to^*_{\mathcal{R}} c'[t'_n]_q$ with reductions parallel to or strictly below q. Since t does not match $c'[t'_n]_q|_{q'}$ and is linear, there must be some reduction at a position $q'.q''$ where $q'' \in Pos_{\mathcal{F}}(t)$ and q'' is either parallel to or below o.



Hence, t is overlapped by some rule of $\mathcal{R}$ at some position parallel to or below o, and we get a contradiction to $\langle t, o, \lambda\rangle \in \Pi_{orth}$. □

Lemma 21 establishes that for a CDP chain candidate it suffices to consider the nested contexts unmodified as long as one only considers patterns from Π_{orth} to check whether the nested contexts forbid their hole position implying that the candidate chain is not an actual chain.

Regarding the second problem of considering a complete set of CDP chain candidates, we present two solutions yielding two concrete CDP processors, that we call "simple context processor" and "context processor (based on tree automata)" respectively.

4.4.2.1 A Simple Context Processor

The idea of the *simple context processor* is to consider only CDP chain candidates of a bounded length n. Assuming a finite set of CDPs, there are only finitely many possible sequences of CDPs of this length. Then, if none of these sequences containing a certain CDP $s \to t[c]$ is an FP-CDP chain (which then cannot be part of an infinite FP-CDP chain, cf. Lemma 22 below) it is sound to delete $s \to t[c]$ from the given CDP problem.

The following lemma establishes that every finite subsequence of CDPs forming an FP-CDP chain form an FP-CDP chain in turn.

Lemma 22. *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be a CDP problem and $\alpha_1, \alpha_2, \dots$ be an FP-CDP chain where $\alpha_i \in \mathcal{P}$ for all $i \geq 1$. Then $\alpha_m, \alpha_{m+1}, \dots, \alpha_{m+n}$ as well as $\alpha_m, \alpha_{m+1}, \dots$ are FP-CDP chains for all $m, n \geq 1$.*

Proof. We consider the original CDP sequence $\alpha_1, \alpha_2, \dots$ and write $\alpha_1 = s_1 \to$

$t_1[c_1], \alpha_2 = s_2 \to t_2[c_2], \ldots$. Since this CDP sequence is an FP-CDP chain, we have

$$\begin{aligned}
s_1\sigma &\to_{\mathcal{P}} c_1[t_1\sigma]_{p_1} = c'_1[t_1\sigma]_{p'_1} \\
\xrightarrow{\not\leq p'_1}{}^{*}_{\mathcal{R}} c''_1[s_2\sigma]_{p'_1} &\to_{\mathcal{P}} c''_1[c_2[t_2\sigma]_{p_2}]_{p'_1} = c'_2[t_2\sigma]_{p'_2} \\
&\ldots \\
\xrightarrow{\not\leq p'_{m-1}}{}^{*}_{\mathcal{R}} c''_{m-1}[s_m\sigma]_{p'_{m-1}} &\to_{\mathcal{P}} c''_{m-1}[c_m[t_m\sigma]_{p_m}]_{p'_{m-1}} = c'_m[t_m\sigma]_{p'_m} \\
\xrightarrow{\not\leq p'_m}{}^{*}_{\mathcal{R}} c''_m[s_{m+1}\sigma]_{p'_m} &\to_{\mathcal{P}} c''_m[c_{m+1}[t_{m+1}\sigma]_{p_{m+1}}]_{p'_m} = c'_{m+1}[t_{m+1}\sigma]_{p'_{m+1}} \\
&\ldots \\
\xrightarrow{\not\leq p'_{m+n-1}}{}^{*}_{\mathcal{R}} c''_{m+n-1}[s_{m+n}\sigma]_{p'_{m+n-1}} &\to_{\mathcal{P}} c''_{m+n-1}[c_{m+n}[t_{m+n}\sigma]_{p_{m+n}}]_{p'_{m+n-1}} \\
&\ldots
\end{aligned}$$

for some substitution σ according to Definition 34. However, according to Lemma 20 we also have

$$\begin{aligned}
s_m\sigma &\to_{\mathcal{P}} c_m[t_m\sigma]_{p_m} = \tilde{c}'_m[t_m\sigma]_{\tilde{p}'_m} \\
\xrightarrow{\not\leq \tilde{p}'_m}{}^{*}_{\mathcal{R}} \tilde{c}''_m[s_{m+1}\sigma]_{\tilde{p}'_m} &\to_{\mathcal{P}} \tilde{c}''_m[c_{m+1}[t_{m+1}\sigma]_{p_{m+1}}]_{\tilde{p}'_m} = \tilde{c}'_{m+1}[t_{m+1}\sigma]_{\tilde{p}'_{m+1}} \\
&\ldots \\
\xrightarrow{\not\leq \tilde{p}'_{m+n-1}}{}^{*}_{\mathcal{R}} \tilde{c}''_{m+n-1}[s_{m+n}\sigma]_{\tilde{p}'_{m+n-1}} &\to_{\mathcal{P}} \tilde{c}''_{m+n-1}[c_{m+n}[t_{m+n}\sigma]_{p_{m+n}}]_{\tilde{p}'_{m+n-1}} \\
&\ldots
\end{aligned}$$

where $\tilde{c}'_i$ (resp. $\tilde{c}''_i$) is obtained from c'_i (resp. c''_i) through extraction, i.e. $\tilde{c}'_i = c'_i|_o$ (resp. $\tilde{c}''_i = c''_i|_o$) for some position o for all $i \in \{m, \ldots, m+n, \ldots\}$. Hence, $\alpha_m, \alpha_{m+1}, \ldots, \alpha_{m+n}$ resp. $\alpha_m, \alpha_{m+1}, \ldots$ are proper FP-CDP chains. □

Using Lemma 22 we get that if no sequence of CDPs of length n involving a certain CDP α is a proper FP-CDP chain, no infinite FP-CDP chain involves α and hence α can be soundly deleted. Thus, by additionally using Lemma 21 we can define an effective CDP processor, the *simple context processor.*

Definition 36 (simple context processor)**.** *Let $Prob = (\{s \to t[c[\Box]_p]\} \uplus \mathcal{P}, \mathcal{R}, \Pi, T)$ be a CDP problem. Given a bound n the simple context processor (SCP_n) returns*

- $\{(\mathcal{P}, \mathcal{R}, \Pi, T)\}$ *if for every sequence of CDPs*

  $$s \to t[c[\Box]_p], s_2 \to t_2[c_2[\Box]_{p_2}], \ldots, s_n \to t_n[c_n[\Box]_{p_n}]$$

  position $p.p_2.\cdots.p_n$ is forbidden in the term $c[c_2[\ldots c_n[erase(t_n)]_{p_n} \ldots]_{p_2}]_p$ by a forbidden pattern of Π_{orth}, and

- $\{Prob\}$ *otherwise.*

Theorem 16. *The CDP processor SCP_n is sound and complete for every $n > 1$.*

Proof. Completeness of the processor is trivial, since either one CDP is deleted or the problem is returned unmodified. In either case infinity of the returned problem implies infinity of the original one.

Regarding soundness assume towards a contradiction that $Prob$ is infinite while $SCP_n(Prob)$ is finite (i.e. the single problem contained in the set of returned problems). If $SCP_n(Prob) = \{Prob\}$, then soundness is trivial. Otherwise, let $Prob = (\{s \to t[c[\Box]_p]\} \uplus \mathcal{P}, \mathcal{R}, \Pi, T)$ and $SCP_n(Prob) = \{(\mathcal{P}, \mathcal{R}, \Pi, T)\}$, i.e. the CDP $s \to t[c[\Box]_p]$ has been deleted by the processor. Since $Prob$ is infinite, there exists an infinite FP-CDP chain $S\colon \alpha_1, \alpha_2, \ldots$ with $\alpha_i \in \{s \to t[c[\Box]_p]\} \uplus \mathcal{P}$ for all $i \geq 1$. Moreover, $s \to t[c[\Box]_p]$ occurs infinitely often in S, since otherwise there would exist an infinite FP-CDP chain without $s \to t[c[\Box]_p]$ starting after the last occurrence of $s \to t[c[\Box]_p]$ in S (using Lemma 22), thus contradicting finiteness of $(\mathcal{P}, \mathcal{R}, \Pi, T)$.

Now consider a subsequence of length $n+1$ of S starting at an occurrence of $s \to t[c[\Box]_p]$, i.e. $\alpha_m, \alpha_{m+1}, \ldots, \alpha_{m+n+1}$. According to Definition 36, since $s \to t[c[\Box]_p]$ has been deleted, the position $p.p_{m+1}.\cdots.p_{m+n+1}$ is forbidden in the term $c[c_{m+1}[\ldots c_{m+n}[erase(t_{m+n})]_{p_{m+n}} \ldots]_{p_{m+1}}]_p$ by a forbidden pattern of Π_{orth} where $c_i[\Box]_{p_i}$ is the context associated to the CDP α_i for all $i \geq 1$ and t_{m+n} is the right-hand side of the CDP α_{m+n}.

Using Lemma 21 we obtain that the same position is Π_{orth}-forbidden in every term obtained from $c[c_{m+1}[\ldots c_{m+n}[erase(t_{m+n})]_{p_{m+n}} \ldots]_{p_{m+1}}]_p$ by reduction parallel to or below $p.p_{m+1}.\cdots.p_{m+n+1}$. Thus, there cannot be a subsequent CDP step at this position and hence $\alpha_m, \alpha_{m+1}, \ldots, \alpha_{m+n+1}$ is not a proper FP-CDP chain. However, by Lemma 22 this implies that S is not a proper FP-CDP chain and we get a contradiction. $\Box$

Example 54. *Consider the CDP problem of Example 53 and let* $n = 3$. *The only candidate CDP sequence of length* 3 *is* α, α, α *where* $\alpha = a^{\#} \to a^{\#}[f(\Box)]$. *The nested context corresponding to this CDP sequence is* $f(f(f(\Box)))$, *the relevant position is* 1.1.1 *and* $f(f(f(erase(a^{\#}))))$ *is* $f(f(f(a)))$. *In this example* $\Pi_{orth} = \Pi$ *and thus we observe that position* 1.1.1 *is forbidden in the term* $f(f(f(a)))$. *According to Theorem 16 it is sound to delete* α, *thus leaving us with an empty set of CDPs. Hence, we conclude finiteness of the original CDP problem.*

Definition 36 requires to consider all sequences of CDPs of a given length n as CDP chain candidates. However, in practice it is not desirable to consider all n-tuples of CDPs, since the number of these tuples combinatorially explodes. To counter this problem, the sequences of CDPs that need to be considered can be obtained from existing DP graph approximations (cf. e.g. [33, 32, 52]). By the definition of the dependency graph, every DP-chain corresponds to a path in this graph and also every FP-CDP chain corresponds to a path in the DP graph and thus also in every (over-)approximation of this graph.

Example 55. *Consider the TRS $\mathcal{R}$ from Example 32 and one forbidden pattern $\Pi = \{\langle x : (y : zs), \epsilon, b\rangle\}$. We have $CDP(\mathcal{R})$ =*

$$\begin{array}{rr}
\{\alpha_1 \colon inf^{\#}(x) \to inf^{\#}(s(x))\ [x : \Box] & \alpha_2 \colon inf^{\#} \to T(x)\ [x : inf(s(\Box))] \\
\alpha_3 \colon inf^{\#} \to T(x)\ [\Box : inf(s(x))] & \alpha_4 \colon 2nd^{\#}(x : y : zs) \to T(y)\ [\Box] \\
\alpha_5 \colon T(inf(x)) \to T(x)\ [inf(\Box)] & \alpha_6 \colon T(inf(x)) \to inf^{\#}(x)\ [\Box]\}
\end{array}$$

Now we apply the simple context processor to the CDP problem $(CDP(\mathcal{R}), \mathcal{R}, \Pi, T)$ with a bound of $n = 3$ and considering the CDP α_1. By computing some DP graph approximation one observes that all CDP chain candidates of length 3 starting with the CDP α_1 are the following:

$\alpha_1, \alpha_2, \alpha_5$	*with corresponding context:*	$x : (x' : inf(s(inf(\Box))))$
$\alpha_1, \alpha_2, \alpha_6$	*with corresponding context:*	$x : (x' : inf(s(\Box)))$
$\alpha_1, \alpha_3, \alpha_5$	*with corresponding context:*	$x : (inf(\Box) : inf(s(x')))$
$\alpha_1, \alpha_3, \alpha_6$	*with corresponding context:*	$x : (\Box : inf(s(x')))$
$\alpha_1, \alpha_1, \alpha_2$	*with corresponding context:*	$x : (x' : (x'' : inf(s(\Box))))$
$\alpha_1, \alpha_1, \alpha_3$	*with corresponding context:*	$x : (x' : (\Box : inf(s(x''))))$
$\alpha_1, \alpha_1, \alpha_1$	*with corresponding context:*	$x : (x' : x'' : \Box)$

Note that CDPs in chain candidates are assumed to be variable disjoint, so the reoccurring variables have been renamed in the example. It is easy to see that the box position is forbidden in all above contexts, hence this position is also forbidden when $\Box$ is substituted by any term $erase(t)$, because $\Box$ does not occur in any forbidden pattern. Thus, none of the CDP chain candidates is a proper FP-CDP chain and thus according to Theorem 16 it is sound to delete α_1.

4.4.2.2 A Context Processor Based on Tree Automata

The second way to effectively check whether all possible CDP chain candidates are not proper FP-CDP chains is by describing the set of nested contexts associated to these chains by a tree automaton. Then, having a finite representation of these in general infinitely many nested contexts, one can check whether the $\Box$-positions are forbidden in all these contexts.

We start with the simpler problem of describing the nested context of a single given sequence of CDPs $s_1 \to t_1[c_1], \ldots, s_n \to t_n[c_n]$. It is straightforward to construct a tree automaton $\mathcal{A}$ accepting (only) the term $c_1[c_2[\ldots c_n[erase(t_n)]\ldots]]$ where all variables are replaced by a new constant. Replacing all variables by one single new constant is justified by the fact that we are interested only in whether this term is matched by another *linear* term (namely a forbidden pattern from Π_{orth}). It is easy to see that if some term s is matched by another linear term t after replacing all variables in s by some constant not occurring in t, then s and every instance of s is matched by t as well.

In order to check whether the nested context described by $\mathcal{A}$ is matched by some forbidden pattern from Π_{orth}, we construct another tree automaton $\mathcal{B}$ that accepts *all* terms that are matched by any forbidden pattern from Π_{orth} (or contain such a term). Then, the idea is roughly to check whether the language accepted by $\mathcal{A}$ is a sublanguage of $\mathcal{B}$ and if that is the case to conclude that the sequence of CDPs is not a proper FP-CDP chain.

In the following, we discuss the construction of the involved automata in more detail, starting with the one that accepts terms matched by some forbidden pattern of Π_{orth}.

The rest of the section is structured as follows:

1. First, we construct an automaton that accepts ground terms containing a fresh unary symbol H only if the position of the occurrence of H is forbidden by a given Π. This automaton is called the *forbidden pattern automaton* and denoted by $FPA(\Pi)$ (cf. Definition 38 and Lemma 27 below).

2. Second, given a finite sequence of CDPs $s_1 \rightarrow t_1\ [c_1], \ldots, s_n \rightarrow t_n\ [c_n]$, a signature $\mathcal{F}$ and a term t we construct an automaton $DPA((c_1 \ldots c_n), \mathcal{F}, t)$ that accepts instances of terms of the shape $c_1[\ldots c_n[t]]$ (cf. Definition 39 below) and show that in case $L(DPA((c_1 \ldots c_n), \mathcal{F}, H(t))) \subseteq L(FPA(\Pi_{orth}))$ the sequence of CDPs is not a proper FP-CDP chain (cf. Theorem 17 below).

3. Third, we show an auxiliary result about arbitrary directed graphs stating that all cycles of such graphs can be finitely characterized by so called *minimal cycle combinations* (MCCs, cf. Lemmata 28 and 29 below).

4. Finally, we show that in a given dependency graph approximation consisting of CDPs, all terms obtained by nesting contexts of CDPs along cyclic paths in the graph are accepted by the *minimal cycle combination automaton* $MCCA(M, t)$ (after replacing $\Box$ by $H(t)$ in the final context; cf. Definition 46 and Theorem 18 below). Thus, we derive an effective method to simplify CDP problems (cf. Definition 47, Theorem 19 and Corollary 8 below).

The Forbidden Pattern Automaton

Starting from a CDP problem $(\mathcal{P}, \mathcal{R}, \Pi, T)$, the signature $\mathcal{F}'$ of the automaton we are going to construct is the signature of $\mathcal{R}$ plus $\{H, A\}$ where H is a new function symbol of arity one and A is a new constant[4].

The role of the symbol H is to indicate the forbidden position in terms matched by forbidden patterns and A is needed, since the automata we use work on ground terms, so we are going to replace variables in the occurring terms by this new constant. In order to effectively construct an automaton accepting only terms where the positions of H-occurrences are forbidden we need a couple of definitions and lemmas first.

[4]Throughout this section H and A are assumed to be a unary function symbol resp. a constant that are fresh for the signatures resp. CDP problems in question.

Definition 37. *A tree automaton $\mathcal{A} = (\mathcal{Q}, \mathcal{F}, \mathcal{Q}_f, \Delta)$ is a quadruple where $\mathcal{Q}$ is a set of states, $\mathcal{F}$ is a signature, $\mathcal{Q}_f \subseteq \mathcal{Q}$ is a set of final states and Δ is a set of transitions of the form*

$$\begin{array}{rcl} f(q_1, \ldots, q_n) & \to & q, \text{ or} \\ q & \to & q' \end{array}$$

where $f \in \mathcal{F}$, $n = ar(f)$ and $q, q', q_1, \ldots, q_n \in \mathcal{Q}$. A term $t \in \mathcal{T}(\mathcal{F}, \emptyset)$ is accepted by $\mathcal{A}$ if $t \to^*_\Delta q$ for some $q \in \mathcal{Q}_f$. The set of all terms accepted by $\mathcal{A}$ is the language *of $\mathcal{A}$ and denoted by $L(\mathcal{A})$.*

Lemma 23 (automata matching terms). *Given a signature $\mathcal{F}$ and a linear term t, it is possible to construct an automaton with a unique end-state accepting exactly the ground terms over $\mathcal{F}$ matched by t.*

Proof. We prove the result by induction on the depth of t. If t is a constant, we construct an automaton having only one state q which is an end state and one transition $t \to q$. If t is a variable, we construct an automaton again having only one state q which is also an end state and containing the transitions $f(q, \ldots, q) \to q$ for all $f \in \mathcal{F}$. Obviously, this automaton accepts exactly all ground terms.

For the induction step consider $t = f(t_1, \ldots, t_n)$. We construct n automata $\mathcal{A}_1, \ldots, \mathcal{A}_n$ accepting all instances of $t_1, \ldots, t_n$ respectively. The states of these automata can be renamed to be pairwise disjoint. Now we construct an automaton $\mathcal{A}$ consisting of the union of all states of $\mathcal{A}_i$ for $1 \leq i \leq n$ and one additional state q which is also the only end-state. Moreover, the set of transitions of $\mathcal{A}$ is the union of the transitions of $\mathcal{A}_i$ for $1 \leq i \leq n$ and one additional transition $f(q_1, \ldots, q_n) \to q$ where q_i is the unique end state of $\mathcal{A}_i$ for all $1 \leq i \leq n$. □

Lemma 24 (subterm closure). *Given a tree automaton $\mathcal{A}$ accepting terms from $L(\mathcal{A})$ and a signature $\mathcal{F}$, an automaton $stc_{\mathcal{F}}(\mathcal{A})$ (subterm closure) can be constructed accepting exactly the terms of the shape $C[s]$ where $s \in L(\mathcal{A})$ and $C \in \mathcal{T}(\mathcal{F} \uplus \{\Box\}, \emptyset)$ (and $C[s]$ does not contain any occurrence of $\Box$).*

Proof. Let $\mathcal{A} = (\mathcal{Q}, \mathcal{F}, \mathcal{Q}_f, \Delta)$ and $\mathcal{F}'$ be the given signature. Then $stc(\mathcal{A}) = (\mathcal{Q}', \mathcal{F} \cup \mathcal{F}', \mathcal{Q}_f, \Delta')$. where $\mathcal{Q}' = \mathcal{Q} \uplus \{q\}$ and q is a new state. Moreover, $\Delta' =$

$$\begin{array}{rl} \Delta & \cup \\ \{f(q, \ldots, q) \to q \mid f \in \mathcal{F}'\} & \cup \\ \{f(q, \ldots, q)[q']_i \to q' \mid f \in \mathcal{F}'^{\geq 1}, i \in \{1, \ldots, ar(f)\}, q' \in \mathcal{Q}_f\} & \end{array}$$

where $\mathcal{F}'^{\geq 1}$ contains all function symbols from $\mathcal{F}'$ having an arity of at least 1. □

Lemma 25 (function containment). *Given a signature $\mathcal{F}$, a linear term t, a set of variables $X \subseteq Var(t)$ and a designated function symbol $H \in \mathcal{F}$, it is possible to construct an automaton with a unique end-state accepting exactly the ground terms over $\mathcal{F}$ matched by t, i.e., $t\sigma$, where $x\sigma$ does not contain H whenever $x \in X$.*

Proof. The proof is analogous to proof of Lemma 23 with the only difference that in the induction base case if $t = x$ and $x \in X$ we use the automaton $\mathcal{A}$ containing the transitions $f(q, \dots, q) \to q$ for all $f \in \mathcal{F} \setminus \{H\}$. □

Lemma 26 (combination of automata)**.** *Let $\mathcal{A}$ and $\mathcal{B}$ be two tree automata accepting the languages $L(\mathcal{A})$ and $L(\mathcal{B})$ respectively. They can be combined into a tree automaton accepting the language $L(\mathcal{A}) \cup L(\mathcal{B})$. We denote this automaton by $\mathcal{A} \cup \mathcal{B}$.*

Proof. Folklore (cf. eg. [17][Theorem 1.3.1]). □

Now we are ready to define an automaton accepting exactly terms containing the new function symbol H at a forbidden position.

Definition 38 (FP automaton)**.** *Let Π be a set of linear forbidden patterns over a signature $\mathcal{F}$ and let $\mathcal{F}' = \mathcal{F} \uplus \{H, A\}$ where H is unary and A is a constant. The* forbidden pattern automaton *$FPA(\Pi)$ is given by*

$$\bigcup_{\pi \in \Pi} \left(\bigcup_{p \in \overline{Pos}^{\Pi}(t)} stc_{\mathcal{F}}(\mathcal{A}_p^{\pi}) \right) \cup stc_{\mathcal{F}}(\widetilde{\mathcal{A}}_o^{\pi})$$

where $\pi = \langle t, o, \lambda \rangle$ and

- *$\mathcal{A}_p^{\pi}$ is the automaton accepting ground terms $t[H(t|_p)]_p\sigma$ (over $\mathcal{F}'$) where $x\sigma$ does not contain an H symbol for all $x \in Dom(\sigma)$, and*
- *$\widetilde{\mathcal{A}}_o^{\pi}$ is the empty automaton (i.e. accepting the empty language) if $\lambda = h$, and if $\lambda = b$, then $\widetilde{\mathcal{A}}_o^{\pi}$ is the automaton accepting terms $t\sigma$ where additionally $x\sigma$ does not contain the symbol H if $x \notin Var(t|_o)$, and $x\sigma$ does not contain the symbol H at the root position if $x = t|_o$.*

Note that in Definition 38 in case $t|_o = x \in Var$ and $\lambda = b$, $\widetilde{\mathcal{A}}_o^{\pi}$ is given by $\bigcup_{f \in \mathcal{F}} \mathcal{B}_f^{\pi}$ where $\mathcal{B}_f^{\pi}$ is the automaton accepting terms $t[f(x_1, \dots, x_{ar(f)})]_o\sigma$ where $x\sigma$ does not contain the symbol H for all $x \in Dom(\Sigma) \setminus \{x_1, \dots, x_{ar(f)}\}$ (i.e. the variables $x_1, \dots, x_{ar(f)}$ are fresh in t). Hence, $FPA(\Pi)$ can effectively be constructed by the Lemmata 25, 24 and 26.

The following lemma shows that with the tree automaton $FPA(\Pi)$ for a given set of forbidden patterns Π, we can describe the set of terms containing the function symbol H at a forbidden position.

Lemma 27. *The automaton $FPA(\Pi)$ accepts a term $C[H(t)]_p$ over the signature $\mathcal{F} \uplus \{H, A\}$ only if the position p is forbidden in $C[t]_p$ by Π.*

Proof. Assume a term $s = C[H(t)]_p$ is accepted such that p is not forbidden in $C[t]_p = s'$. Hence, s is accepted either by the automaton $stc_{\mathcal{F}}(\mathcal{A}_q^{\pi})$ for some $\pi = \langle u, o, \lambda \rangle \in \Pi$ and some $q \in \overline{Pos}^{\Pi}(u)$, or it is accepted by the automaton $stc_{\mathcal{F}}(\widetilde{\mathcal{A}}_o^{\pi})$. for some $\pi = \langle u, o, \lambda \rangle \in \Pi$.

In the first case, where s is accepted by $stc_{\mathcal{F}}(\mathcal{A}^{\pi}_{q})$, s is of the form $C'[u[H(u|_o)]_o\sigma]_{p'}$ for some position $o \in \overline{Pos}^{\Pi}(u)$ according to Definition 38 and Lemmata 24 and 25. Moreover, C', u, and $x\sigma$ do not contain the symbol H for all $x \in Dom(\sigma)$. Hence, the occurrence of H is unique in s and we have $p = p'.o$. Position $p'.o = p$ is forbidden in $C'[u\sigma]_{p'}$by π. Thus, we obtain a contradiction to our assumption of p being allowed in s'.

In the second case s is accepted by $stc_{\mathcal{F}}(\widetilde{\mathcal{A}}^{\pi}_{o})$. We distinguish two cases. First, let $t|_o \notin Var$. Hence, s is of the form $C'[u\sigma]_{p'}$ (according to Definition 38 and Lemmata 24 and 25), $\lambda = b$ and $p > p'.o$, because C', u and $x\sigma$ do not contain the symbol H unless $x \in Var(u|_o)$ and $u|_o \notin Var$. Hence, p is forbidden in s' by π.

Finally, let $t|_o = x \in Var$. Then, s is of the form $C'[u\sigma]_{p'}$ (according to Definition 38 and Lemmata 24 and 25), $\lambda = b$ and C', u and $x\sigma$ do not contain the symbol H unless $x = u|_o$. Moreover, if $u|_o\sigma$ contains H-symbols, then they occur below the root. Hence, $p > p'.o$ and thus p is forbidden in s'. □

Example 56. *Consider a set of forbidden patterns containing the pattern*

$$\pi_1 \colon \langle f(f(x)), 1, b\rangle$$

For the automaton we use the signature $\{f, g, H, A\}$ f, g and H are unary and A is a constant. The automaton $stc(\mathcal{A}^{\pi_1}_{1.1})$ consists of the transitions

$$\begin{array}{rcl} h(q_1) & \rightarrow & q_1 \text{ for all } h \in \{f, g\} \\ A & \rightarrow & q_1 \\ H(q_1) & \rightarrow & q_2 \\ f(q_2) & \rightarrow & q_3 \\ f(q_3) & \rightarrow & q_4 \\ h(q_4) & \rightarrow & q_4 \text{ for all } h \in \{f, g\} \end{array}$$

where q_4 is the unique end state. The automaton $stc(\widetilde{\mathcal{A}}^{\pi_1}_{1})$ consists of the transitions

$$\begin{array}{rcl} h(q_1) & \rightarrow & q_1 \text{ for all } h \in \{H, f, g\} \\ A & \rightarrow & q_1 \\ f(q_1) & \rightarrow & q_2 \\ f(q_2) & \rightarrow & q_3 \\ h(q_3) & \rightarrow & q_3 \text{ for all } h \in \{f, g\} \end{array}$$

where q_3 is the unique end state.

The automaton $FPA(\Pi)$ contains the following transitions:

$$\begin{array}{rcl}
A & \to & q_1 \\
A & \to & q_1' \\
h(q_1) & \to & q_1 \textit{ for all } h \in \{f, g\} \\
H(q_1) & \to & q_1' \\
h(q_1') & \to & q_1' \textit{ for all } h \in \{f, g, H\} \\
f(q_1) & \to & q_2 \\
f(q_1') & \to & q_2 \\
f(q_2) & \to & q_3 \\
h(q_3) & \to & q_4 \textit{ for all } h \in \{f, g\}
\end{array}$$

The unique end state is q_4. Note that $FPA(\Pi)$ is not deterministic.

An Automaton for Nested Contexts w.r.t. CDP Sequences

Next, we discuss the construction of an automaton accepting terms obtained by nesting contexts of subsequent contextual dependency pairs in sequences of dependency pairs where additionally the (new) function symbol H occurs at the hole position of the context. This construction basically relies on the construction of an automaton that accepts terms matched by a given term t, which has already been established in Lemma 23.

Definition 39 (CDP automaton)**.** *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be an FP-CDP problem where $\mathcal{R} = (\mathcal{F}, R)$ and let $S\colon s_1 \to t_1, [c_1] \dots, s_n \to t_n[c_n]$ be a sequence of CDPs. The CDP automaton $DPA((c_1, \dots, c_n), \mathcal{F}, t_n)$ corresponding to this CDP chain is the automaton accepting ground terms of the shape $c_1'[\dots c_n'(t_n)]\sigma$ where $x\sigma \in \mathcal{T}(\mathcal{F} \cup \{A\}, \emptyset)$ for all $x \in Var(t_n)$ and c_i' is obtained from c_i by replacing the variables of c_i by the constant A.*

Replacing variables in contexts by new constants is justified by the fact that whenever some linear forbidden pattern matches a context after replacing variables by the new constant, every instance of the context (before substituting the new constant) is matched as well.

Theorem 17 (analyzing nested contexts)**.** *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be an FP-CDP problem with $\mathcal{R} = (\mathcal{F}, R)$ and let $S\colon s_1 \to t_1, [c_1] \dots, s_n \to t_n[c_n]$ be a sequence of dependency pairs. If $L(DPA((c_1, \dots, c_n), \mathcal{F}, H(t_n))) \subseteq L(FPA(\Pi_{orth}))$, then S cannot be part of an infinite FP-CDP chain.*

Proof. $L(DPA((c_1, \dots, c_n), \mathcal{F}, H(t_n))) \subseteq L(FPA(\Pi))$ means by Definition 39 and Lemma 27 that in every instance of a term $t = c_1'[c_2'[\dots c_n'[t_n]_{p_n} \dots]_{p_2}]_{p_1}$ (where c_i' is obtained from c_i by replacing all variables by the new constant A) position $p_1.p_2.\cdots.p_n$ is forbidden by Π.

Now assume towards a contradiction that S is part of an infinite CDP chain. Then we have

$$\begin{array}{rcl}
c^0[s_1]_p\sigma & \to_{\mathcal{P}} & c^0[c_1[t_1\sigma]_{p_1}]_p = c^1[t_1\sigma]_{p^1} \\
\overset{\not\leq p^1}{\to}{}^*_{\mathcal{R}}\ \tilde{c}^1[s_2\sigma]_{p^1} & \to_{\mathcal{P}} & \tilde{c}^1[c_2[t_2\sigma]_{p_2}]_{p^1} = c^2[t_2\sigma]_{p^2} \\
\overset{\not\leq p^2}{\to}{}^*_{\mathcal{R}}\ \tilde{c}^2[s_3\sigma]_{p^2} & \to_{\mathcal{P}} & \tilde{c}^2[c_3[t_3\sigma]_{p_3}]_{p^2} = c^3[t_3\sigma]_{p^3} \\
\overset{\not\leq p^3}{\to}{}^*_{\mathcal{R}}\ \dots & \to_{\mathcal{P}} & \dots \\
 & \vdots & \\
\overset{\not\leq p^{n-1}}{\to}{}^*_{\mathcal{R}}\ \tilde{c}^{n-1}[s_n\sigma]_{p^{n-1}} & \to_{\mathcal{P}} & \tilde{c}^{n-1}[c_n[t_n\sigma]_{p_n}]_{p^{n-1}} = \tilde{c}^n[t_n\sigma]_{p^n}.
\end{array}$$

for some context c^0. Moreover, we have $c^0[c_1[c_2[\dots c_n[t_n\sigma]_{p_n}\dots]_{p_2}]_{p_1}]_p \to^*_{\mathcal{R}} \tilde{c}^n[t_n\sigma]_{p^n}$ with reductions parallel to p or p_i for some $1 \leq i \leq n$. Hence, we also have $c^{0\prime} \to^*_{\mathcal{R}} c^{0\prime\prime}$ and $c'_i \to^*_{\mathcal{R}} c''_i$ for all $1 \leq i \leq n$ (where $c^{0\prime}$ is obtained from c^0 by replacing all variables by A) such that $\tilde{c}^n[t_n\sigma]_{p^n} = c^{0\prime\prime}[c''_1[\dots c''_n[t_n\sigma]]]$ (after replacing variables in $c^n[t_n\sigma]_{p^n}$ by A). Thus , Lemma 21 is applicable and we obtain that position p^n is forbidden in $\tilde{c}^n[t_n\sigma]_{p^n}$. Moreover, again by Lemma 21, position p^n is forbidden in every term s obtained from $\tilde{c}^n[t_n\sigma]_{p^n}$ by $\mathcal{R}$-reduction at positions parallel to or strictly below p^n. Hence, we get a contradiction to the existence of another CDP after $s_n \to t_n[c_n]$ in the infinite CDP chain. □

Theorem 17 enables us to check whether a given sequence of CDPs is an FP-CDP chain. However, we did not yet tackle the problem of effectively computing a complete set of candidate CDP sequences in order to soundly deduce non-existence of any infinite FP-CDP chain.

In order to address this problem we are going to construct a tree automaton accepting the language of *all* possible nested contexts corresponding to candidate CDP sequences. To this end we start with the observation that every FP-CDP chain corresponds to a path in the dependency graph (as defined e.g. in [9]), that we obtain when we strip the contextual pairs of their contexts and consider them as ordinary dependency pairs. Hence, we start by describing infinite paths in general graphs in a convenient way.

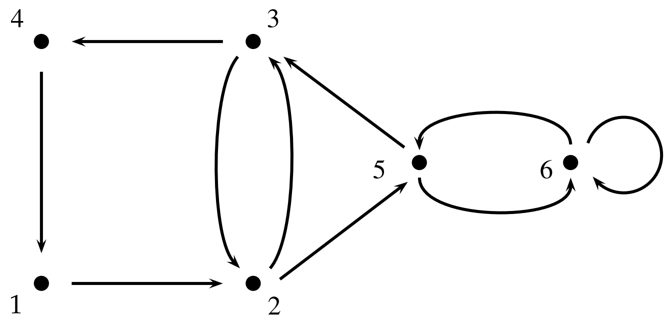
Characterizing Cycles in Graphs by Minimal Cycle Combinations

First, we provide the definition of cycle and minimal cycles in a graph:

Definition 40 (minimal cycle). *A cycle for a node n_1 in a graph is a finite sequence of nodes $(n_1, n_2, \dots, n_m)$ such that there is an edge from n_i to n_{i+1} for all $1 \leq i < m$, $n_1 = n_m$ and $n_1 \notin \{n_2, n_3, \dots, n_{m-1}\}$. Moreover, the cycle is minimal if the nodes $n_1, \dots, n_{m-1}$ are pairwise distinct.*

By *node*(C) we denote the node of the cycle C, i.e. the first and last node of the sequence of nodes.

Example 57. *Consider the graph*



which contains the following minimal cycles:

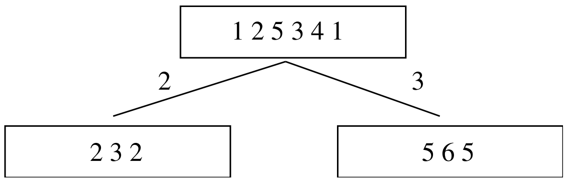
12341	*232*	*2532*
565	*6*	*125341*

A key observation of this section is that we can describe the in general infinite set of cycles of a graph finitely as combinations of minimal cycles. These combinations are represented as trees and called *minimal cycle combinations.*

Definition 41 (minimal cycle combinations)**.** *Let $\mathcal{G}$ be a (finite) graph and let $C_1, \ldots, C_n$ be the set of minimal cycles of $\mathcal{G}$. A* minimal cycle combination *(MCC) is a tree whose nodes are minimal cycles and whose edges $C_i - C_j$ are labeled by a positive integer k from $\{1, \ldots, |C_i|\}$ if C_j is a cycle for the k^{th} node of C_i. A minimal cycle combination is* hierarchical *if node$(C_i) \notin C_j$ whenever C_j is below C_i.*

We say an MCC *is for node n* if its root is a cycle for node n. Note that the depth of a hierarchical MCC is bounded by the number of nodes in the graph $\mathcal{G}$.

Example 58. *Consider the graph $\mathcal{G}$ from Example 57. The following is an example of an MCC for $\mathcal{G}$:*



The root node of the tree is a cycle for node 1*, hence the MCC is for node* 1*. The children of the root node are minimal cycles for the nodes* 2 *and* 5 *respectively. Since these minimal cycles do not contain the node* 1*, the MCC is hierarchical.*

An important feature of MCCs is that they describe a potentially infinite set of concrete cycles.

Definition 42 (cycles associated to MCCs)**.** *Let M be a hierarchical MCC for a node n. The set $Cyc(M)$ of cycles for this MCC is inductively defined by $root(M)$ if M consists only of the root node. Otherwise, let k be the minimal label of any edges adjacent to the root and let M' be the MCC obtained from M by omitting* *all direct subtrees $M_1, \ldots, M_m$ connected to the root by edges labelled with k. Then $Cyc(M) = \{c[c']_k \mid c \in Cyc(M'), c' \in Seq((\overline{Cyc}(M_1) \cup \ldots \cup \overline{Cyc}(M_m)))\}$ where $Seq(A)$ describes the set of all (finite) sequences of elements from A, $\overline{Cyc}(M) = \{(n_1, n_2, \ldots, n_l) \mid (n_1, n_2, \ldots, n_l, n_1) \in Cyc(M)\}$, $Seq(\emptyset) = \epsilon$ and $c[c']_k$ means that in the cycle c the cycle c' is pasted before the k^{th} element.*

Note that choosing the minimal label k in Definition 42 ensures that the index k remains valid in cycles from $Cyc(M')$. These cycles differ from $root(M)$ only at index positions greater than k.

Example 59. *Consider the hierarchical MCC M of Example 58. The smallest index of any edge adjacent to the root node is 2. Hence, we compute $\overline{Cyc}(M')$ where M' is obtained from M by removing this edge and the corresponding child of the root node (i.e. the cycle (232)). For M' there is only one edge adjacent to the root node having index 3. We have*

$$Cyc(M') = \{c[c']_3 \mid c = root(M), c' \in C_{56}\}$$

where $C_{56} = Seq(\{(56)\}) = \{\epsilon, 56, 5656, 565656, \ldots\}$. Hence,

$$Cyc(M') = \{1\ 2\ c_{56}\ 5\ 3\ 4\ 1 \mid c_{56} \in C_{56}\}.$$

Now $Cyc(M) = \{c[c']_2 \mid c \in Cyc(M'), c' \in C_{23}\}$ where $C_{23} = Seq(\{(23)\}) = \{\epsilon, 23, 2323, 232323, \ldots\}$. Hence, we have

$$Cyc(M) = \{1\ c_{23}\ 2\ c_{56}\ 5\ 3\ 4\ 1 \mid c_{23} \in C_{23}, c_{56} \in C_{56}\}.$$

Lemma 28. *For a finite graph $\mathcal{G}$ there is only a finite number of hierarchical MCCs having different sets of associated cycles.*

Proof. We use induction on the (finite) depth of MCCs. The number of MCCs of depth 1 (i.e. the ones consisting of only one (root) node), is the number of minimal cycles in $\mathcal{G}$ which is finite, since no minimal cycle can be longer than the number of nodes plus one by definition.

Now consider an MCC of depth n. The induction hypothesis yields that there are only finitely many MCCs of depth $n-1$ (say the number is m). Moreover, there are only finitely many labels (since the root node is a minimal cycle; say the number is l), hence each MCC of depth n either has at most $m*l$ immediate sub-MCCs or it contains identical children (of the root node) connected by edges with identical labels. If the root node has two identical children (subtrees) connected by identical labels, one of these subtrees can be erased without changing the associated cycles, because of idempotency (and associativity, commutativity) of $\cup$ (cf. Definition 42). Hence, there are only finitely many MCCs of depth n having different sets of associated cycles. □

The next lemma shows that every cycle is described by a hierarchical MCC.

Lemma 29. *Let C be a cycle. Then there is a hierarchical MCC M such that $C \in Cyc(M)$.*

Proof. First, if C is minimal, then M consists only of one node C. If C is not minimal we prove the result by induction on the length of C. If $|C| = 1$, then C is minimal. Otherwise, assume the length of C is $n > 1$ and C is not minimal. Let $n_i \neq n_1$ be the first node in C such that there exists $n_j = n_i$ with $j > i$ and $n_k \neq n_i$ for $i < k < j$. Then let $C' = (n_i, n_{i+1}, \dots, n_j)$ and $C'' = (n_1, \dots, n_{i-1}, n_i, n_{j+1}, \dots, n_m)$ both have lengths smaller than n and thus by the induction hypothesis there are hierarchical MCCs M' and M'' such that $C' \in Cyc(M'$)respectively $C'' \in Cyc(M'')$. We obtain M by adding the tree M' as child to the root of M'' with label i and get $C \in Cyc(M)$ by Definition 42. □

According to Lemmata 28 and 29 we can describe all cycles of a graph by a finite set of hierarchical MCCs. In the following, we will use this result to describe all possible sequences of CDPs w.r.t. DP graph approximations.

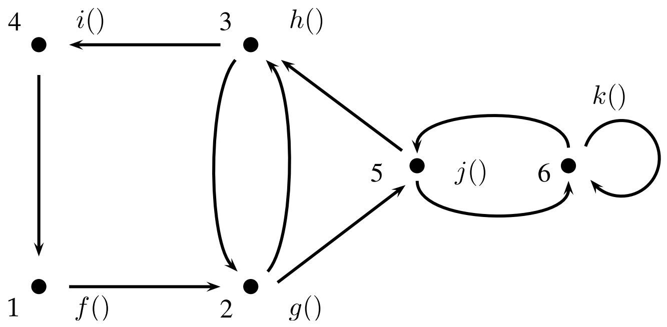
An Automaton Accepting Contexts of CDPs Corresponding to Cyclic DP Graph Paths

From now on we are exclusively concerned with dependency graphs, i.e. graphs whose nodes are CDPs. Since we are only interested in the context component of the CDPs, we will sometimes consider graphs where the nodes are labelled by the contexts only for notational simplicity. In a first step we are going to describe nested contexts obtained by cycles of CDPs in a DP graph.

Definition 43 (DP cycle context)**.** *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be an FP-CDP problem and let $S\colon s_1 \to t_1\ [c_1] \dots, s_n \to t_n\ [c_n]$ be a sequence of CDPs in some DP graph approximation. The associated overall context $ctx(S)$ is $c_1[c_2[\dots c_n]]$. Moreover, given a hierarchical minimal cycle combination M we write $ctxs(M)$ for the set $\{ctx(s_1 \to t_1\ [c_1] \dots, s_n \to t_n\ [c_n]) \mid \exists C' \in Cyc(M), C' = s_1 \to t_1\ [c_1], \dots, s_n \to t_n\ [c_n], s_1 \to t_1\ [c_1]\}$.*

Note that for given cycles $n_1, \dots, n_k, n_1$ in a DP graph we are actually interested in the contexts corresponding to the sequence $n_1, \dots, n_k$ of nodes. The reason is that these sequences (where the last node of the cycle is omitted) can be concatenated to form valid paths in the graph, while for instance the sequence $n_1, \dots, n_k, n_1, n_1, \dots, n_k, n_1$ might not be a proper path if there is no edge from n_1 to n_1.

Example 60. *We consider the graph of Example 57 where now contexts are attached to each node:*



For the minimal cycle 12341 we have $ctx(1234) = f(g(h(i(\Box))))$. Moreover, we have e.g.

$$f(g(h(g(j(k(j(k(j(h(i(\Box))))))))))) \in ctxs(M)$$

for the hierarchical MCC M of Example 58. Note that the given context corresponds to the cycle

$$123256565341.$$

Abusing notation we write $ctxs(M)[t]$ for the set of terms given by $\{c[t] \mid c \in ctxs(M)\}$.

The crucial result of this section is that given a hierarchical MCC M in a DP graph and a term t we can construct a (finite) tree automaton $MCCA(M, t)$ accepting all terms from $ctxs(M)[H(t)]$.

To construct this automaton we need some machinery. First, we define the function *CycAut*, which, given automata $\mathcal{A}_1, \ldots, \mathcal{A}_m$ each accepting a language of contexts, computes an automaton $CycAut(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ that accepts (among others) contexts $c_1[c_2[\ldots c_n[\Box] \ldots]]$ for arbitrary n where $c_i \in L(\mathcal{A}_j)$ for some $j \in \{1, \ldots, m\}$ and all $i \in \{1, \ldots, n\}$ (cf. also Lemma 32 below).

Definition 44 (CycAut)**.** *Given tree automata $\mathcal{A}_1, \ldots, \mathcal{A}_m$ (with disjoint sets of states) each having a unique end state and $\Box$ in their signature, $CycAut(\mathcal{A}_1, \ldots, \mathcal{A}_m)$ is obtained from $\bigcup_{1 \le i \le m} \mathcal{A}_i$ as follows. The end states of all $\mathcal{A}_i$ are unified to a state q_{end} and for each $1 \le i \le m$ and each $1 \le j \le l_i$ a transition $q_{end} \to q_i^j$ is added where $\{q_i^1, \ldots, q_i^{l_i}\}$ are the states of $\mathcal{A}_i$ such that $\mathcal{A}_i$ contains a transition $\Box \to q_i^j$. Moreover, a transition $\Box \to q_{end}$ is added.*

Example 61. *Consider two automata $\mathcal{A}$ and $\mathcal{B}$ where $\mathcal{A}$ consists of the transitions*

$$\begin{array}{rcl} \Box & \to & q_1 \\ A & \to & q_2 \\ f(q_1, q_2) & \to & q_3 \end{array}$$

and q_3 is the (unique) end state of $\mathcal{A}$, and $\mathcal{B}$ consists of the transitions

$$\begin{array}{rcl} \Box & \to & q_1 \\ g(q_1) & \to & q_2 \end{array}$$

where q_2 is the (unique) end state of $\mathcal{B}$. $\mathcal{A}$ accepts (only) the term $f(\Box, A)$ and $\mathcal{B}$ accepts (only) the term $g(\Box)$. Then $CycAut(\mathcal{A}, \mathcal{B})$ has the transitions

$$\begin{array}{rcl} \Box & \to & q_1 \\ A & \to & q_2 \\ f(q_1, q_2) & \to & q_{end} \\ \Box & \to & q_1' \\ g(q_1') & \to & q_{end} \\ q_{end} & \to & q_1 \\ q_{end} & \to & q_1' \end{array}$$

and q_{end} is the unique end state of $CycAut(\mathcal{A}, \mathcal{B})$. $CycAut(\mathcal{A}, \mathcal{B})$ accepts terms from the following set of terms

$$\{g(\Box), f(\Box, A), f(g(\Box), A), g(f(\Box, A)), g(g(\Box)), f(f(\Box, A), A), \ldots\}$$

Next, we define the automaton $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$ that accepts terms of $L(\mathcal{A})$ where occurrences of the variable a have been replaced by terms from $L(\mathcal{B})$.

Definition 45 (concatenation of automata)**.** *Let $\mathcal{A} = (\mathcal{Q}^{\mathcal{A}}, \mathcal{F}^{\mathcal{A}}, \mathcal{Q}_f^{\mathcal{A}}, \Delta^{\mathcal{A}})$ and $\mathcal{B} = (\mathcal{Q}^{\mathcal{B}}, \mathcal{F}^{\mathcal{B}}, \mathcal{Q}_f^{\mathcal{B}}, \Delta^{\mathcal{B}})$ be tree automata (where $\mathcal{Q}^{\mathcal{A}} \cap \mathcal{Q}^{\mathcal{B}} = \emptyset$) and let a be a constant of $\mathcal{F}^{\mathcal{A}}$. By $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$ we denote the tree automaton $(\mathcal{Q}, \mathcal{F}, \mathcal{Q}_f, \Delta)$ where*

$$\begin{array}{rcl} \mathcal{Q} & = & \mathcal{Q}^{\mathcal{A}} \cup \mathcal{Q}^{\mathcal{B}} \\ \mathcal{F} & = & \mathcal{F}^{\mathcal{A}} \cup \mathcal{F}^{\mathcal{B}} \\ \mathcal{Q}_f & = & \mathcal{Q}_f^{\mathcal{A}} \\ \Delta & = & \Delta^{\mathcal{B}} \cup \Delta'^{\mathcal{A}} \end{array}$$

and $\Delta'^{\mathcal{A}} = \{l \to r \mid l \to r \in \Delta^{\mathcal{A}}, l \neq a\} \cup \{q \to r \mid l \to r \in \Delta^{\mathcal{A}}, l = a, q \in \mathcal{Q}_f^{\mathcal{B}}\}$.

Note that we consider $\overset{a}{\rightsquigarrow}$ to be left-associative. Hence, parenthesis are omitted in complex expressions containing $\overset{a}{\rightsquigarrow}$.

Example 62. *Consider tree automata $\mathcal{A}$ and $\mathcal{B}$ where $\mathcal{A}$ consists of the transitions*

$$\begin{array}{rcl} a & \to & q_1 \\ b & \to & q_2 \\ g(q_2) & \to & q_3 \\ f(q_1, q_3) & \to & q_4 \end{array}$$

and has the (unique) end state q_4. Hence, $\mathcal{A}$ accepts (only) the term $f(a, g(b))$. $\mathcal{B}$ consists of the transitions

$$\begin{array}{rcl} c & \to & q_1 \\ h(q_1) & \to & q_2 \end{array}$$

and has the (unique) end state q_2. It accepts (only) the term $h(c)$. The automaton $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$ is given by the transitions

$$\begin{array}{rcl} q_2' & \to & q_1 \\ b & \to & q_2 \\ g(q_2) & \to & q_3 \\ f(q_1, q_3) & \to & q_4 \\ c & \to & q_1' \\ h(q_1') & \to & q_2'. \end{array}$$

Note that the states of $\mathcal{B}$ were renamed to be disjoint from the states of $\mathcal{A}$. $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$ accepts (only) the term $f(h(c), g(b))$.

Lemma 30. *Let $\mathcal{A}$ and $\mathcal{B}$ be tree automata. and let a be a constant of the signature of $\mathcal{A}$. If a term t is obtained by replacing each occurrence of a in a term $t' \in L(\mathcal{A})$ by some terms $s \in L(\mathcal{B})$, then t is accepted by $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$.*

Proof. By construction of $\mathcal{A} \overset{a}{\rightsquigarrow} \mathcal{B}$. □

The following construction defines the tree automaton $MCCA(M, t)$ that accepts contexts (where the hole is replaced by some term t) obtained by paths of the DP graph forming cycles from $Cyc(M)$. This is the crucial step of describing the infinite set of these cycles (and associated nested contexts) finitely to finally obtain an effective termination criterion.

Definition 46 (automaton for MCC)**.** *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be a forbidden pattern dependency pair problem with $\mathcal{R} = (\mathcal{F}, R)$ and M be a hierarchical MCC for a CDP (i.e. node) $s \to t\ [c]$ in some DP graph approximation. The automaton $MCCA(M, t)$ is inductively defined as follows.*

Let $M_k^1, \ldots, M_k^{l_k}$ be the immediate subtrees of M connected to the root of M by an edge with label k and let $(m_1 \ldots m_n)$ be the minimal cycle of the root of M. Moreover, let c_i be the context associated to the node (i.e. CDP) m_i for all $1 \leq i \leq n$ in the DP graph approximation. Then $MCCA(M, t)$ is given by

$$\begin{array}{ll} & DPA((c_1), \mathcal{F}, \Box) \\ \overset{\Box}{\rightsquigarrow} & CycAut(MCCA(M_2^1, \Box), \ldots, MCCA(M_2^{l_2}, \Box)) \\ \overset{\Box}{\rightsquigarrow} & DPA((c_2), \mathcal{F}, \Box) \\ \ldots & \\ \overset{\Box}{\rightsquigarrow} & CycAut(MCCA(M_{n-1}^1, \Box), \ldots, MCCA(M_{n-1}^{l_{n-1}}, \Box)) \\ \overset{\Box}{\rightsquigarrow} & DPA((c_{n-1}), \mathcal{F}, H(t)) \end{array}$$

Example 63. *Consider the graph of Example 60 and the MCC M of Example 58. For ease of readability we provide only the transitions of the following automata.*

They all have a unique end state denoted by q_{end}. Let $\mathcal{F} = \{f, g, h, i, j, k\}$ and $t = f(x)$ be. We have

$$
\begin{aligned}
DPA(f(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_1 \\ f(q_1) & \to & q_{end} \end{array} \right. \\
DPA(g(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_2 \\ g(q_2) & \to & q_{end} \end{array} \right. \\
DPA(h(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_3 \\ h(q_3) & \to & q_{end} \end{array} \right. \\
DPA(i(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_4 \\ i(q_4) & \to & q_{end} \end{array} \right. \\
DPA(j(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_5 \\ j(q_5) & \to & q_{end} \end{array} \right. \\
DPA(k(\Box), \mathcal{F}, \Box) &= \left\{ \begin{array}{rcl} \Box & \to & q_6 \\ k(q_6) & \to & q_{end} \end{array} \right. \\
DPA(i(\Box), \mathcal{F}, H(t)) &= \left\{ \begin{array}{rcl} A & \to & q_7 \\ l(q_7) & \to & q_7 \;\; \textit{for all } l \in \{f, g, h, i, j, k\} \\ f(q_7) & \to & q_8 \\ H(q_8) & \to & q_9 \\ i(q_9) & \to & q_{end} \end{array} \right.
\end{aligned}
$$

M_2 consists only of one (root) node. Hence, $MCCA(M_2, \Box)$ is given by

$$DPA(g(\Box), \mathcal{F}, \Box) \overset{\Box}{\rightsquigarrow} DPA(h(\Box), \mathcal{F}, \Box)$$

and consists of the transitions

$$
\begin{array}{rcl}
\Box & \to & \tilde{q}_3 \\
h(\tilde{q}_3) & \to & \tilde{q}_3' \\
\tilde{q}_3' & \to & \tilde{q}_2 \\
g(\tilde{q}_2) & \to & q_{end}.
\end{array}
$$

Analogously, $MCCA(M_5, \Box)$ is given by

$$DPA(j(\Box), \mathcal{F}, \Box) \overset{\Box}{\rightsquigarrow} DPA(k(\Box), \mathcal{F}, \Box)$$

and consists of the transitions

$$
\begin{array}{rcl}
\Box & \to & \tilde{q}_6 \\
k(\tilde{q}_6) & \to & \tilde{q}_6' \\
\tilde{q}_6' & \to & \tilde{q}_5 \\
j(\tilde{q}_5) & \to & q_{end}.
\end{array}
$$

$CycAut(MCCA(M_2, \Box))$ resp. $CycAut(MCCA(M_5, \Box))$ can be obtained from the automaton $MCCA(M_2, \Box)$ resp. $MCCA(M_5, \Box)$ by adding the transition $q_{end} \to \tilde{q}_3$ resp. $q_{end} \to \tilde{q}_6$ as well as the transition $\Box \to q_{end}$ (to both). Hence, we have all ingredients to construct $MCCA(M, t)$ given by

$$\begin{aligned}
& DPA((f(\Box)), \mathcal{F}, \Box) \\
\overset{\Box}{\rightsquigarrow}\ & CycAut(MCCA(M_2, \Box)) \\
\overset{\Box}{\rightsquigarrow}\ & DPA((g(\Box)), \mathcal{F}, \Box) \\
\overset{\Box}{\rightsquigarrow}\ & CycAut(MCCA(M_5, \Box)) \\
\overset{\Box}{\rightsquigarrow}\ & DPA((j(\Box)), \mathcal{F}, \Box) \\
\overset{\Box}{\rightsquigarrow}\ & DPA((h(\Box)), \mathcal{F}, \Box) \\
\overset{\Box}{\rightsquigarrow}\ & DPA((i(\Box)), \mathcal{F}, H(t)).
\end{aligned}$$

It consists of the following transitions:

$$\begin{array}{rclrcl}
A & \to & q_7 & l(q_7) & \to & q_7 \text{ for all } l \in \{f, g, h, i, j, k\} \\
f(q_7) & \to & q_8 & H(q_8) & \to & q_9 \\
i(q_9) & \to & q_9' & q_9' & \to & q_3 \\
h(q_3) & \to & q_3' & q_3' & \to & q_5 \\
j(q_5) & \to & q_5' & q_5' & \to & \tilde{q}_6 \\
k(\tilde{q}_6) & \to & \tilde{q}_6' & \tilde{q}_6' & \to & \tilde{q}_5 \\
j(\tilde{q}_5) & \to & \tilde{q}_5' & \tilde{q}_5' & \to & q_5' \\
q_5' & \to & q_2 & g(q_2) & \to & q_2' \\
q_2' & \to & \tilde{q}_3 & h(\tilde{q}_3) & \to & \tilde{q}_3' \\
\tilde{q}_3' & \to & \tilde{q}_2 & g(\tilde{q}_2) & \to & \tilde{q}_2' \\
\tilde{q}_2' & \to & q_2' & q_2' & \to & q_1 \\
f(q_1) & \to & q_{end} & & &
\end{array}$$

In the following two lemmas we prove that any automaton $MCCA(M, \Box)$ (where M is some MCC) accepts terms containing exactly one occurrence of $\Box$ and that

$$CycAut(MCCA(M_1, \Box), \ldots, MCCA(M_m, \Box))$$

accepts arbitrary combinations of contexts accepted by some $MCCA(M_i, \Box)$ (obtained by nesting).

Lemma 31. *Let $(\mathcal{P}, \mathcal{R}, \Pi, T)$ be an FP-CDP problem with $\mathcal{R} = (\mathcal{F}, R)$ and M be a hierarchical MCC for a CDP (i.e. node) $s_1 \to t_1\ [c_1]$ in some DP graph approximation. Every term accepted by $MCCA(M, \Box)$ contains the constant $\Box$ exactly once.*

Proof. Given a context (with unique $\Box$ position) c, the automaton $DPA((c), \mathcal{F}, \Box)$ accepts only terms with exactly one occurrence of $\Box$ by Definition 39. Hence, for a proof by structural induction that $MCCA(M, \Box)$ accepts only terms with exactly one occurrence of $\Box$, according to Definition 46 it suffices to show that

- whenever two automata $\mathcal{A}$ and $\mathcal{B}$ accept only terms containing exactly one $\Box$, then so does $\mathcal{A} \overset{\Box}{\rightsquigarrow} \mathcal{B}$; and
- whenever tree automata $\mathcal{A}_1, \dots, \mathcal{A}_k$ accept only terms containing exactly one $\Box$, then so does $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$.

For $\mathcal{A} \overset{\Box}{\rightsquigarrow} \mathcal{B}$ assume a term s is accepted that does not contain any $\Box$ symbol. Since terms accepted by $\mathcal{A}$ contain a $\Box$ symbol and thus by Definition 45 a subterm of s must be accepted by $\mathcal{B}$, we get a contradiction to our assumption that $\mathcal{B}$ accepts only terms containing a $\Box$ symbol.

Second, assume a term s containing two or more $\Box$ symbols is accepted by $\mathcal{A} \overset{\Box}{\rightsquigarrow} \mathcal{B}$. According to Lemma 30, s is of the form $s[s_1, \dots, s_l]_{p_1,\dots,p_l}$ where s_i is accepted by $\mathcal{B}$ for all $1 \leq i \leq l$ and $s[\Box, \dots, \Box]_{p_1,\dots,p_n}$ is accepted by $\mathcal{A}$. Now if more than one occurrence of $\Box$ in s are inside s_i for some $1 \leq i \leq l$, then we get a contradiction to $\mathcal{B}$ accepting only terms with exactly one $\Box$-occurrence. On the other hand, if $l > 1$ or there are additional $\Box$-occurrence in s, then we get a contradiction to $\mathcal{A}$ accepting only terms having exactly one occurrence of $\Box$.

Now consider $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$. Let s be a term containing no $\Box$-position that is accepted by $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$. By Definition 44, s must contain a subterm accepted by $\mathcal{A}_i$ for some $1 \leq i \leq k$ or must be $\Box$ itself. In the latter case we get an immediate contradiction. In the former case, by induction, the subterm of s accepted by some $\mathcal{A}_i$ must contain exactly one $\Box$, thus we get a contradiction.

Second, assume s contains two or more occurrences of $\Box$ and is accepted by $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$. Moreover, assume that no proper subterm of s contains two or more occurrences of $\Box$ and is accepted by $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$ (thus s is a minimal counterexample). According to Definition 44 (i.e. by the construction of $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$), s is of the form $s[s_1, \dots, s_l]_{p_1,\dots,p_l}$ where s_i is accepted by $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_k)$ for all $1 \leq i \leq l$ and $s[\Box, \dots, \Box]_{p_1,\dots,p_l}$ is accepted by $\mathcal{A}_i$ for some $1 \leq i \leq k$. By minimality of s, s_i contains at most one occurrence of $\Box$ for all $1 \leq i \leq l$. Hence, since s contains more than one $\Box$, either $l > 1$ or s contains additional occurrences of $\Box$ at some positions parallel to p_i for all $1 \leq i \leq l$. Either way we get a contradiction to $\mathcal{A}_i$ accepting the term $s[\Box, \dots, \Box]_{p_1,\dots,p_l}$, which contains more than one $\Box$ symbol. $\Box$

Lemma 32. *Given automata $\mathcal{A}_1, \dots, \mathcal{A}_n$ accepting terms (i.e. contexts) containing exactly one occurrence of $\Box$, $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ accepts arbitrary terms of the shape $c_{i_1}[c_{i_2}[\dots c_{i_l}[\Box]]]$ where $i_j \in \{0, 1, \dots, n\}$ for all $j \in \{1, \dots, l\}$ and $c_k \in L(A_k)$ for all $1 \leq k \leq n$.*

Proof. For brevity we denote $\mathit{CycAut}(\mathcal{A}_1, \dots, \mathcal{A}_n)$ just by CycAut. We prove the result by induction on l. If l is 0, then the result follows from Definition 44, since there is a transition $\Box \rightarrow q_{end}$ where q_{end} is an end state of CycAut.

Otherwise, consider a term t of the form $c_{i_1}[c_{i_2}[\dots c_{i_l}[\Box]]]$. The induction hypothesis yields that $c_{i_2}[\dots c_{i_l}[\Box]]$ is accepted by CycAut. Moreover, $c_{i_1}[\Box]$ is accepted by *CycAut* (as it is accepted by $\mathcal{A}_{i_1}$). Hence, $c_{i_1}[c_{i_2}[\dots c_{i_l}[\Box]]]$ itself is accepted by *CycAut*, since for every transition $\Box \to q$, there is also a transition $q_{end} \to q$ in *CycAut* according to Definition 44. $\Box$

The next theorem shows the crucial step of describing the (possibly) infinite set $ctxs(M)[H(t)]$ finitely by $MCCA(M,t)$ for some hierarchical MCC M and term t.

Theorem 18. *Let* $(\mathcal{P}, \mathcal{R}, \Pi, T)$ *be an FP-CDP problem with* $\mathcal{R} = (\mathcal{F}, R)$ *where all variables in the contexts of CDPs have been replaced by the new constant* A*, and let* M *be a hierarchical MCC for a CDP (i.e. node)* $s_1 \to t_1\ [c_1]$ *in some DP graph approximation. Every ground instance (over* $\mathcal{F} \cup \{A\}$*) of a term from* $ctxs(M)[H(t)]$ *is accepted by* $MCCA(M,t)$*.*

Proof. Let the root of the MCC M be the minimal cycle $C = (s_1 \to t_1\ [c_1] \dots s_n \to t_n\ [c_n], s_1 \to t_1\ [c_1])$. We prove the result by induction on the size (number of nodes) of M. If M consists only of the root, then $ctxs(M)[H(t)] = \{c_1[c_2[\dots c_n[H(t)]]]\}$ whose ground instances are accepted by $MCCA(M,t) =$

$$\begin{array}{ll} & DPA((c_1), \mathcal{F}, \Box) \\ \overset{\Box}{\leadsto} & DPA((c_2), \mathcal{F}, \Box) \\ \dots & \\ \overset{\Box}{\leadsto} & DPA((c_n), \mathcal{F}, H(t)) \end{array}$$

according to Definition 39 and Lemma 30.

For the induction step let M' be the MCC obtained from M by omitting all children of the root node connected by an edge with label k where k is the minimum label of all edges connected to the root in M. Clearly, the size of M' is smaller than M. Hence, ground instances of terms of $ctxs(M')[H(t)]$ are accepted by $MCCA(M',t)$ by the induction hypothesis.

According to Definitions 42 and 43 every ground instance of a term from the set $ctxs(M)[H(t)]$ is obtained from a ground term $c_1[c_2[\dots c_k[\dots c_m[H(t)\sigma]\dots]\dots]]$ being an instance of a term from $ctxs(M')[H(t)]$ by replacing the context $c_k[\Box]$ by some context $c'_k[c_k[\Box]]$ where c'_k is a context of the set

$$\{ctx(C) \mid C \in Seq(\overline{Cyc}(M_1), \dots, \overline{Cyc}(M_l))\}$$

and $M_1, \dots, M_l$ are the MCCs connected to the root of M with label k. According to Lemma 32, the context c'_k is accepted by

$$CycAut(MCCA(M_1, \Box), \dots, MCCA(M_l, \Box))$$

.

Hence, the term $c_1[c_2[\dots c'_k[c_k[\dots c_m[H(t)\sigma]\dots]]\dots]]$ is accepted by the automaton $MCCA(M,t) =$

$$\begin{array}{ll} & DPA((c_1),\mathcal{F},\Box) \\ \overset{\Box}{\rightsquigarrow} & DPA((c_2),\mathcal{F},\Box) \\ \dots & \\ \overset{\Box}{\rightsquigarrow} & DPA((c_{k-1}),\mathcal{F},\Box) \\ \overset{\Box}{\rightsquigarrow} & CycAut(MCCA(M_1,\Box),\dots,MCCA(M_l,\Box)) \\ \overset{\Box}{\rightsquigarrow} & DPA((c_k),\mathcal{F},\Box) \\ \dots & \\ \overset{\Box}{\rightsquigarrow} & DPA((c_n),\mathcal{F},H(t)) \end{array}$$

□

Lemma 33. *Let $(\mathcal{P},\mathcal{R},\Pi,T)$ be an FP-CDP problem with $\mathcal{R}=(\mathcal{F},R)$ and M be a hierarchical MCC for a CDP (i.e. node) $s\to t\ [c]$ in some DP graph approximation D. Let $C\in Cyc(M)$ be a cycle in D and $C=(s\to t[c]\ s_{i_1}\to t_{i_1}[c_{i_1}]\dots s_{i_n}\to t_{i_n}[c_{i_n}]\ s\to t[c])$. $L(DPA((c\,c_{i_1}\dots c_{i_n}),\mathcal{F},H(t)))\subseteq L(MCCA(M,t))$.*

Proof. Direct consequence of Definition 43 and Theorem 18. □

The following theorem is the basis of a dependency pair processor relying on contexts of CDPs.

Theorem 19. *Let $Prob=(\{s\to t\ [c]\}\uplus\mathcal{P},\mathcal{R},\Pi,T)$ be an FP-CDP problem with $\mathcal{R}=(\mathcal{F},R)$. If for every hierarchical MCC M for $s\to t\ [c]$ in some DP graph approximation (where in contexts of CDPs variables have been replaced by the new constant A) $L(MCCA(M),t)\subseteq L(FPA(\Pi_{orth}))$, then $Prob$ is finite if and only if $(\mathcal{P},\mathcal{R},\Pi,T)$ is finite.*

Proof. The "only if" part is trivial. For the "if" part assume the contrary. Then there exists an infinite FP-CDP chain S w.r.t. *Prob* that contains infinitely many occurrences of $s\to t\ [c]$. Two arbitrary consecutive appearances of $s\to t\ [c]$ in S form a cycle C in the DP graph approximation. According to Theorem 17, $L(DPA((c_1,\dots,c_n),\mathcal{F},H(t)))\nsubseteq L(FPA(\Pi_{orth}))$ where $(c_1\dots c_n)$ are the contexts of the nodes in C. However, according to Lemma 29, there exists an MCC M for $s\to t\ [c]$ such that $C\in Cyc(M)$. Hence, we have $L(DPA((c_1,\dots,c_n),\mathcal{F},H(t)))\subseteq L(MCCA(M,t))$ according to Lemma 33 and as $L(MCCA(M,t))\subseteq L(FPA(\Pi_{orth}))$ we get a contradiction. □

Note that the condition of Theorem 19 can effectively be checked in practice due to Lemma 28.

Building on Theorem 19 we define the context processor based on tree automata.

Definition 47 (context processor)**.** *Let $Prob = (\{s \to t[c[\Box]_p]\} \uplus \mathcal{P}, \mathcal{R}, \Pi, T)$ be a CDP problem. The context processor CP returns*

- $\{(\mathcal{P}, \mathcal{R}, \Pi, T)\}$, *if for every hierarchical MCC M for $s \to t\ [c[\Box]_p]$ in some DP graph approximation $L(MCCA(M), t) \subseteq L(FPA(\Pi_{orth}))$, and*
- $\{Prob\}$ *otherwise.*

Corollary 8. *The context processor CP is sound and complete.*

Benchmarks
We implemented the CDP framework and the context processors (i.e. the simple context processor and the context processor based on tree automata) in the termination tool VMTL (cf. Chapter 5 below)[5]. In order to evaluate the practical power of this approach we tested this implementation on the TRSs of the outermost category of the TPDB[6].Since outermost rewriting is a special case of rewriting with forbidden patterns (in particular, rewriting with forbidden b-patterns), the CDP approach is applicable to these systems. In our test run 291 TRS were evaluated, 158 of which were proven to be outermost terminating in the termination competition 2008 ([1]). Table 4.1 shows the results of VMTL on the test set. At the time of writing, VMTL does not support non-termination analysis of outermost TRSs. Hence, Table 4.1 indicates only the positive results of VMTL and various other termination tools tested on the same set of examples. We cite the results of the termination competition 2008, since the then most powerful tool (regarding positive termination proofs) Jambox did not participate in the subsequent years. These other tools are

- AProVE ([31]), which proves outermost termination by transforming TRSs such that (innermost) termination implies outermost termination of the original TRS. The transformations used are the ones from Raffelsieper et. al. ([75]) and Thiemann ([84]).
- TrafO ([75]), which proves outermost termination by transformations either, using the transformation of Raffelspieler et. al. ([75]) and analyzing the resulting TRSs with Jambox ([24]).
- "Jambox goes out", which proves outermost termination by transforming TRS into context-sensitive TRSs, such that their termination implies outermost termination of the original TRS (cf. [25]).

TTT2 ([53]) participated in the outermost category of the termination competition 2008 but was specialized (exclusively) on disproving outermost termination. Hence, the results of TTT2 are omitted in Table 4.1.

[5]However, at the time of writing this thesis these implementations are only prototypical and not yet contained in the publicly available version of VMTL.

[6]The termination problem database, available at `http://termcomp.uibk.ac.at/`

VMTL Simple	**VMTL**	**AProVE**	**TrafO**	**Jambox goes out**
33	60	27	46	72

Table 4.1: Number of successful outermost termination proofs of various systems.

VMTL was run in two modes. The first one, denoted “VMTL Simple” in Table 4.1, did not use the context processor CP but only the simple context processor SCP_n with $n = 3$ for the analysis of contexts. The second mode was full VMTL (denoted “VMTL” in Table 4.1) which additionally used the CP. The use of CP in the termination analysis approximately doubles the power of VMTL on the test set. However, VMTL is outperformed by “Jambox goes out” which won the category.

The reasons for VMTL being less powerful than “Jambox goes out” are twofold. First, the use of structural dependency pairs adds significant complexity to the initial CDP problems. In particular, for some outermost terminating TRSs where VMTL failed to find an outermost termination proof, we observed that the simplified CDP problems obtained at the end of failed proof attempts by VMTL consisted of structural CDPs only (in their first component). The second reason for the lack of power of VMTL compared to Jambox is the use of Π_{orth} in the context processors. By excluding certain forbidden patterns in the context analysis performed by CP or SCP_n the power is reduced.

Addressing both of these problems appears to be an interesting and promising way to improve the CDP framework and make it even more competitive in the future.

4.5 Computing Useful Results via Rewriting with Forbidden Patterns

In this section we investigate the relation of normal forms obtained under reduction with a TRS $\mathcal{R}$ and normal forms obtained under reduction with $(\mathcal{R}, \Pi)$ for some set of forbidden patterns Π. An important application of restricted rewriting is to compute (parts) of results w.r.t. the underlying unrestricted systems in a more efficient or otherwise operationally superior way (e.g. while some rewrite relation is weakly normalizing, a suitable restriction could be strongly normalizing, cf. e.g. Example 34). Hence, we are mainly interested in criteria that imply a certain connection between normal forms obtained by restricted and unrestricted rewriting. In our case this connection will be that normal forms of restricted rewriting are root-stable w.r.t. unrestricted rewriting.

We are going to use canonical context-sensitive rewriting as defined in [55, 60] as an inspiration for our approach. There, for a given (left-linear) rewriting system $\mathcal{R}$ certain restrictions on the associated replacement map μ guarantee that $\to_\mu$-normal forms are $\to_\mathcal{R}$-head-normal-forms. Hence, results computed by $\to_\mu$ and $\to_\mathcal{R}$ share the same root symbol.

The basic idea is that reductions that are essential to create a more outer redex should not be forbidden. In the case of context-sensitive rewriting this is guaranteed by demanding that whenever an f-rooted term t occurs (as subterm) in the left-hand side of a rewrite rule and has a non-variable direct subterm $t|_i$, then $i \in \mu(f)$.

It turns out that for rewriting with forbidden patterns severe restrictions on the shape of the patterns are necessary in order to obtain results similar to the ones for canonical context-sensitive rewriting in [55, 60]. First, no forbidden patterns of the shape $\langle \_, \epsilon, h\rangle$ or $\langle \_, \_, a\rangle$ may be used as they are in general not compatible with the desired root-normalizing behaviour of our forbidden pattern rewrite system.

Moreover, for each pattern $\langle t, p, \_\rangle$ we demand that

- t is linear,
- p is a variable or maximal (w.r.t. to the prefix ordering $\leq$ on positions) non-variable position in t, and
- for each position $q \in Pos(t)$ with $q||p$ we have $t|_q \in V$.

We call the class of patterns obtained by the above restrictions *simple patterns.*

Definition 48 (simple patterns)**.** *Let* Π *be a set of forbidden patterns. The subset* Π_s *of* simple *forbidden patterns of* Π *is given by those forbidden patterns* $\langle t, p, \lambda\rangle$ *that satisfy*

- $\lambda = b$*, or both* $\lambda = h$ *and* $p > \epsilon$*; and*
- t *is linear; and*
- $t|_p \in Var$ *or* $t|_p = f(x_1, \dots, x_{ar(f)})$ *for some function symbol* f*; and*
- *for each position* $q \in Pos(t)$ *with* $q||p$ *we have that* $t|_q$ *is a variable.*

We say that Π is a set of simple forbidden patterns if $\Pi = \Pi_s$. Basically, these syntactical properties of forbidden patterns are necessary to ensure that reductions which are essential to enable other, more outer reductions are not forbidden. Moreover, these properties, contrasting those introduced in Definition 49 below, are independent of any concrete rewrite system.

The forbidden patterns of the TRS $(\mathcal{R}, \Pi)$ in Example 64 below are not simple, since the patterns contain terms with parallel non-variable positions. This is the reason, why it is not possible to head-normalize terms (w.r.t $\mathcal{R}$) with $\to_\Pi$:

Example 64. *Consider the TRS* $\mathcal{R}$ *given by*

$$f(b, b) \to g(f(a, a)) \qquad a \to b$$

and forbidden patterns $\langle f(a, a), 1, h\rangle$ *and* $\langle f(a, a), 2, h\rangle$*.* $f(a, a)$ *is linear and* 1 *and* 2 *are maximal positions (w.r.t.* $\leq$*) within this term. However, positions* 1 *and* 2 *are both non-variable and thus e.g. for* $\langle f(a, a), 1, h\rangle$ *there exists a position* $2||1$ *such that* $f(a, a)|_2 = a \notin V$*. Hence,* Π *is too restrictive to compute all* $\mathcal{R}$*-head-normal forms* *in this example. Indeed, $f(a,a) \to_{\mathcal{R}}^{*} f(b,b) \to_{\mathcal{R}} g(f(a,a))$ where the latter term is a $\mathcal{R}$-head-normal form.*

The term $f(a,a)$ is a Π-normal form, although it is not a head-normal form (w.r.t. $\mathcal{R}$). Note also that the (first components of) forbidden patterns are not unifiable with the left-hand side of the rule that is responsible for the (later) possible root-step when reducing $f(a,a)$, not even if the forbidden subterms in the patterns are replaced by fresh variables.

Now we are ready to define canonical rewriting with forbidden patterns within the class of simple forbidden patterns. To this end, we demand that patterns do not overlap with left-hand sides of rewrite rules in a way such that reductions necessary to create a redex might be forbidden.

Definition 49 (canonical forbidden patterns)**.** *Let $\mathcal{R} = (\mathcal{F}, R)$ be a TRS with simple forbidden patterns $\Pi_{\mathcal{F}}$ (w.l.o.g. we assume that R and $\Pi_{\mathcal{F}}$ have no variables in common). Then, $\Pi_{\mathcal{F}}$ is $\mathcal{R}$-*canonical *(or just* canonical*) if the following holds for all rules $l \to r \in R$:*

1. *There is no pattern (t, p, λ) such that*
   - *$t'|_q$ and l unify for some $q \in O_{\mathcal{F}}(t)$ where $t' = t[x]_p$ and $q > \epsilon$, and*
   - *there exists a position $q' \in O_{\mathcal{F}}(l)$ with $q.q' = p$ for $\lambda = h$ respectively $q.q' > p$ for $\lambda = b$.*
2. *There is no pattern (t, p, λ) such that*
   - *t' and $l|_q$ unify for some $q \in O_{\mathcal{F}}(l)$ where $t' = t[x]_p$, and*
   - *there exists a position q' with $q.q' \in O_{\mathcal{F}}(l)$ and $q' = p$ for $\lambda = h$ respectively $q' > p$ for $\lambda = b$.*

Here, x denotes a fresh variable.

Example 65. *Consider the TRS $\mathcal{R}$ given by the single rule*

$$l = f(g(h(x))) \quad \to \quad x = r\,.$$

Then, $\Pi = \{\langle t, p, h\rangle\}$ with $t = g(f(a))$, $p = 1.1$ is not canonical, since $t[y]_p|_q = g(f(y))|_1 = f(y)$ and l unify where $q = q' = 1$ and thus $q.q' = p$ (hence $root(l|_{q'}) = g$). Moreover, also $\Pi = \{\langle t, p, h\rangle\}$ with $t = g(i(x))$, $p = 1$ is not canonical, since $l|_q = g(h(x))$ and $t[y]_p = g(y)$ unify for $q = 1$ and $q.p = 1.1$ is a non-variable position in l.

On the other hand, $\Pi = \{\langle g(g(x)), 1.1, h\rangle\}$ is canonical. Note that all of the above patterns are simple.

In order to prove that normal forms obtained by rewriting with simple and canonical forbidden patterns are actually head-normal forms w.r.t. unrestricted rewriting, and also to provide more intuition on canonical rewriting with forbidden patterns, we define the notion of a *partial redex* (w.r.t. to a rewrite system $\mathcal{R}$) as a term that is matched by a non-variable term l' which in turn matches the left-hand side of some rule of $\mathcal{R}$. We call l' a *witness* for the partial match.

Definition 50 (partial redex)**.** *Given a rewrite system $\mathcal{R} = (\mathcal{F}, R)$, a* partial redex *is a term s that is matched by a non-variable term l' which in turn matches the left-hand side of some rule in R. The (non-unique) term l' is called* witness *for a partial redex s.*

Thus, a partial redex can be viewed as a candidate for a future reduction step, which can only be performed if the redex has actually been created through more inner reduction steps. Hence, the idea of canonical rewriting with forbidden patterns could be reformulated as guaranteeing that the reduction of subterms of partial redexes is allowed whenever these reductions are necessary to create an actual redex.

Lemma 34. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a* left-linear *TRS with* canonical *(hence, in particular,* simple*) forbidden patterns $\Pi_\mathcal{F}$. Moreover, let s be a partial redex w.r.t. to the left-hand side of some rule l with witness l' such that $l|_p \notin V$ but $l'|_p \in V$. Then in the term $C[s]_q$ the position $q.p$ is allowed by $\Pi_\mathcal{F}$ for reduction provided that q is allowed for reduction.*

Proof. Assume on the contrary that $q.p$ is forbidden in $C[s]_q$. As position q is allowed, this means that there is a forbidden pattern $\langle t, o, \lambda\rangle$, such that $\lambda \in \{h, b\}$ and t matches $C[s]_q$ at some position q' and $q < q'.o \leq q.p$. Assume $\lambda = b$. As s partially matches l, we have that $root(s|_{p'}) = root(l|_{p'})$ for all $p' < p$. Hence, as all positions parallel to p are variable positions in t (due to simplicity of Π) and t is linear, we have that either $t|_{o'}$ unifies with $l[x]_p$ for some position o' such that $o'.p > o$ (if $q' \leq q$), or t unifies with $l[x]_p|_{o'}$ such that $p > o'.o$ (if $q' > q$). Either way, we get a contradiction to the canonicity of Π (cf. Definition 49). The case where $\lambda = h$ is analogous. □

Theorem 20. *Let $\mathcal{R} = (\mathcal{F}, R)$ be a* left-linear *TRS with* canonical *(hence, in particular,* simple*) forbidden patterns $\Pi_\mathcal{F}$. Then $\to_{\mathcal{R},\Pi_\mathcal{F}}$-normal forms are $\to_\mathcal{R}$-head-normal forms.*

Proof. For a proof by minimal counterexample assume s is an $\to_{\mathcal{R},\Pi_\mathcal{F}}$-normal form, but not a $\to_\mathcal{R}$-head-normal form, and has minimal depth.

If the depth of s is 0, then it is either a constant or a variable. In case it is a variable, it is a $\to_\mathcal{R}$-head-normal form. Otherwise, if it is a constant and not $\to_\mathcal{R}$-head-normal, then it is not a $\to_{\mathcal{R},\Pi_\mathcal{F}}$-normal form, because only patterns of the shape $\langle \_, \epsilon, h\rangle$ can forbid root reduction steps and these are not simple (cf. Definition 48).

Note that s cannot be an $\mathcal{R}$-redex itself, because in this case it would also be $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible, as there are no $\langle \_, \epsilon, h\rangle$-patterns.

Now assume the depth of s is greater than 0. Since the term s is not a $\rightarrow_{\mathcal{R}}$-head-normal form, there exists a reduction sequence $S\colon s \xrightarrow{>\epsilon}{}^*_{\mathcal{R}} t = l\sigma$. Hence, s is a partial redex and there is some maximal subterm $s|_p$ of s where $p \in Pos_{\mathcal{F}}(l)$ that is not an $\rightarrow_{\mathcal{R}}$-head-normal form (otherwise s would be a redex because of left-linearity of $\mathcal{R}$). According to the minimality of s, $s|_p$ must be $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible.

Thus, as s is not $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible, there must be some forbidden pattern $\langle t, o, \lambda\rangle$, where t matches s at some position $q < p$ and forbids the reduction of some position $q.o \geq p$ (because of Lemma 34). We distinguish two cases. First, if $s|_p$ is a redex, then s is $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible, because position p cannot be forbidden in s according to Lemma 34.

Second, if $s|_p$ is not a redex, then it is a partial redex w.r.t. to some $l \rightarrow r \in R$ and contains a maximal proper subterm $s|_{p'}$ ($p' > p$) which is $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible and not a $\rightarrow_{\mathcal{R}}$-head-normal form. Again position p' cannot be forbidden in s according to Lemma 34. Thus, again either $s|_{p'}$ is a redex implying $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducibility of s or it contains a $\rightarrow_{\mathcal{R},\Pi_{\mathcal{F}}}$-reducible proper subterm $s|_{q''}$.

Eventually, either an allowed redex in s is found or there is some subterm $s|_{p^{(n)}}$ of s such that $|p^{(n)}| - |p| > n$ where n is the maximal term depth of all forbidden patterns. Thus, s is reducible below $p^{(n)}$ if and only if $s|_p$ is reducible below o where $p.o = p^{(n)}$. Since $s|_p$ is reducible below o (because by our construction this reduction step is necessary to head-normalize $s|_p$), we have a contradiction to s being irreducible w.r.t. $\rightarrow_{\Pi}$. □

Given a left-linear and confluent rewrite system $\mathcal{R}$ and a set of canonical forbidden patterns Π such that $\rightarrow_{\Pi}$ is well-founded, one can thus normalize a term s (provided that s is normalizing) by computing the $\rightarrow_{\Pi}$-normal form t of s which is $\mathcal{R}$-root-stable according to Theorem 20, and then do the same recursively for the immediate subterms of t. Confluence of $\mathcal{R}$ assures that the unique normal form of s will indeed be computed this way.

Example 66. *As the forbidden pattern defined in Example 33 is (simple and) canonical, Theorem 20 yields that $\rightarrow_{\mathcal{R},\delta}$-normal forms are $\rightarrow_{\mathcal{R}}$-head-normal forms. For instance we get $2nd(inf(0)) \rightarrow_{\Pi}^{*} s(0)$.*

Example 67. *Consider the TRS with $\mathcal{R}$ and forbidden patterns Π from Example 34. We will prove below that $\mathcal{R}$ is Π-terminating (cf. Example 45). Furthermore, we show that every well-formed ground term that is reducible to a normal form in $\mathcal{R}$ is reducible to the same normal form with $\rightarrow_{\mathcal{R},\Pi}$ and that every $\rightarrow_{\mathcal{R},\Pi}$-normal form is root-stable w.r.t. $\rightarrow_{\mathcal{R}}$.*

Proof. Regarding root-stability of $\rightarrow_{\mathcal{R},\Pi}$-normal forms, assume on the contrary that there is a non-root-stable $\rightarrow_{\mathcal{R},\Pi}$ normal-form s of minimal term depth. Since s is non-root-stable, $root(s) \in \{app, from, take\}$. The immediate subterms of s are reducible (in $\mathcal{R}$) to terms $t_1, \ldots, t_n$ such that eventually s becomes a redex. Each t_i

(for $1 \leq i \leq n$) is rooted by a constructor, because the left-hand sides of rules in $\mathcal{R}$ are patterns. Hence (because of the shape of the left-hand sides of $\mathcal{R}$), if s is not a redex, then some immediate subterm of s is not root-stable. Moreover, this subterm is also a $\rightarrow_{\mathcal{R},\Pi}$-normal form as no forbidden pattern term in Π has a defined root symbol and thus, the immediate subterms of s are allowed w.r.t. Π. Thus, we have a contradiction to minimality of s.

Regarding the power of $\rightarrow_{\mathcal{R},\Pi}$ to compute $\mathcal{R}$-normal forms, assume on the contrary that there is a well-formed ground Π-normal form s that is reducible to an $\mathcal{R}$-normal form $t \neq s$ in $\mathcal{R}$, and that s has minimal depth among all such terms.

First, note that no well-formed ground $\mathcal{R}$-normal form t can contain a defined symbol, as all functions are completely defined over ground constructor arguments of the respective types (otherwise, any subterm of t rooted by some innermost defined symbol would have to be reducible, thus contradicting $\mathcal{R}$-irreducibility of t).

Let $f = root(s|_p)$ be an outermost defined symbol in s. First, f cannot be $from$, as in this case s would not be $\rightarrow_{\mathcal{R}}$-normalizing. Second, assume $f = root(s|_p) = take$. As $take$ is not part of any forbidden pattern term, the immediate subterms of $s|_p$ must be $\rightarrow_{\mathcal{R},\Pi}$-normal forms and thus root-stable. Hence, as $s|_p$ must eventually be reducible in some $\mathcal{R}$-reduction and all immediate subterms of $s|_p$ are root-stable (and $root(s|_p)$ is an outermost defined symbol in s) $s|_p$ itself must be a redex and we get a contradiction to s being a $\rightarrow_{\mathcal{R},\Pi}$ normal form, because no $take$-rooted redex is forbidden by Π.

Finally, assume $f = app$. We distinguish two cases. First, assume $s|_p$ is not forbidden for reduction by Π. Then either, $s|_p$ is an app-rooted $\rightarrow_{\mathcal{R},\Pi}$-normal form that is reducible in $\mathcal{R}$ to an $\mathcal{R}$-normal form (remember that there are no defined symbols in s above p) which is a contradiction (as this $\mathcal{R}$-normal form must be rooted by a constructor and $s|_p$ can thus not be root-stable). Otherwise, $s|_p$ has the form $app(from(s_1), s_2)$ which is non-normalizing, hence we get a contradiction to our assumption of s being $\mathcal{R}$-normalizable.

Second, assume $s|_p$ is forbidden in s. thus $s = C[s_1 : app(s_2 : app(s_3, s_4), s_5)]_q$ where $q.2 = p$ and $root(C|_o)$ is a constructor for all $o \leq q$. In this case we take a closer look at the inner app-term, i.e. at $s|_{p.2}$. Again this subterm could be forbidden by Π or allowed. We investigate the general case of having several nested app-symbols, i.e. where s has the shape $C[s_1 : app(s_2 : app(s_3 : app(s_4 : \ldots app(s_n, s'_n)), \ldots), s'_4), s'_3), s'_2)]$ and s_n is not matched by $x : app(y, z)$. Thus $app(s_n, s'_n)$ is not forbidden for reduction by Π. Either s_n is rooted by $from$ in which case it is easy to see that s is not $\mathcal{R}$-normalizing, or s_n is a $\rightarrow_{\mathcal{R},\Pi}$ normal form in which case it must be rooted by : or nil, because it must be reducible to a :-rooted term or nil in $\mathcal{R}$ in a normalizing reduction of s, since the app-symbol cannot be erased. In the latter case s would not be a $\rightarrow_{\mathcal{R},\Pi}$ normal form, as the innermost (indicated) app term would be reducible, and we would have a contradiction. $\Box$

4.6 Automated Synthesis of Suitable Forbidden Patterns.

In this section we are going to utilize the machinery of Section 4.4.2, and in particular the simple context processor SCP_n, in order to synthesize suitable forbidden patterns for a given rewrite system $\mathcal{R}$. The basic idea is to construct the CDPs of $\mathcal{R}$ assuming an empty set of forbidden patterns Π and then by an analysis with the SCP_n processor synthesize the forbidden patterns needed to ensure Π-termination of $\mathcal{R}$ on the fly.

More precisely, we analyze nested contexts obtained by sequences of CDPs of bounded length (as in Definition 36). Let $c_1[\dots[c_n[erase(t_n)]_{p_n}\dots]_{p_1}$ be a term obtained by this nested context analysis. In order to successfully apply the SCP_n processor, position $p_1.\cdots.p_n$ must be forbidden in this term. Hence, we synthesize a forbidden pattern $\langle c_1[\dots[c_n[erase(t_n)]_{p_n}\dots]_{p_1}, p_1.\cdots.p_n, h\rangle$, that forbids exactly this position. By doing this for every sequence of CDPs of length n starting with the CDP corresponding to the context c_1, this CDP can be soundly deleted according to Theorem 16 provided that the generated forbidden patterns are in Π_{orth}. However, forbidden patterns obtained this way might not be orthogonal to the rewrite system and thus not be in Π_{orth}. In order to overcome this problem, terms in the first component of synthesized forbidden patterns can be "generalized", i.e. linearized and subterms at positions where overlaps with the rule system occur can be replaced by fresh variables. By doing this the rewrite relation becomes more restrictive (since the patterns match object terms more easily). Moreover, since the patterns after this generalization are orthogonal to $\mathcal{R}$, the SCP_n processor is applicable on the fly for simplifying the termination problems.

We provide an algorithmic schema for the forbidden pattern synthesis:

1. Compute $CDP(\mathcal{R})$ assuming an empty Π.

2. Choose some CDP $s_1 \to t_1\ [c_1]$.

3. For all CDP sequences

$$s_1 \to t_1[c_1[\Box]_{p_1}], s_2 \to t_2[c_2[\Box]_{p_2}], \dots, s_n \to t_n[c_n[\Box]_{p_n}]$$

   (a) If position $p_1.\cdots.p_n$ is allowed in $c_1[\dots[c_n[erase(t_n)]_{p_n}\dots]_{p_1}$,

      i. Create a forbidden pattern $\langle c_1[\dots[c_n[erase(t_n)]_{p_n}\dots]_{p_1}, p_1.\cdots.p_n, h\rangle$.

      ii. Generalize $\langle c_1[\dots[c_n[erase(t_n)]_{p_n}\dots]_{p_1}, p_1.\cdots.p_n, h\rangle$ so that it is orthogonal to $\mathcal{R}$, obtaining $\langle u, o, \lambda\rangle$.

      iii. Add $\langle u, o, \lambda\rangle$ to Π.

4. Delete the CDP $s_1 \to t_1\ [c_1]$ and continue the Π-termination analysis (e.g. at Stage 2).

Example 68. *Consider a CDP problem $(\mathcal{P}, \mathcal{R}, \Pi, T)$ where*

$$\begin{array}{rcl} \mathcal{P} & = & \{a^\# \to a^\#[f(\Box)]\} \\ \mathcal{R} & = & \{a \to f(a)\} \\ \Pi & = & \emptyset \end{array}$$

(cf. also Example 53). An SCP_2 processor encounters e.g. the term

$$f(f(erase(a^\#)) = f(f(a)).$$

Thus, a forbidden pattern $\pi = \langle f(f(a)), 1.1, h\rangle$ could be used. This forbidden pattern is orthogonal to $\mathcal{R}$, hence there is no need to generalize it. Indeed, when this forbidden pattern is used, there is no infinite FP-CDP chain.

Usually one wants to restrict the shape of the generated patterns for instance by demanding that all forbidden patterns contain allowed redexes and do not overlap (each other), in order to ensure that Π-normal forms are normal forms (w.r.t. $\mathcal{R}$); then termination of $\to_\Pi$ implies weak termination of $\to_{\mathcal{R}}$.

A second choice for restrictions on the shape of forbidden patterns might be *canonical* forbidden patterns as defined the previous section.

Synthesis of forbidden patterns adhering to these syntactical restrictions can be done analogously to the way patterns orthogonal to $\mathcal{R}$ are synthesized. Namely, by generalizing the forbidden patterns to make them compatible with syntactical constraints immediately after their creation.

Example 69. *Consider the TRS of $\mathcal{R}$ of Example 32 and the contextual dependency pair*

$$inf^\#(x) \to inf^\#(s(x))[x : \Box]$$

and an empty set of forbidden patterns. Applying an SCP_2 processor we get a term $x : (x' : inf(s(x')))$, which needs to be generalized, since it is not linear and thus not canonical (because not simple) and not in Π_{orth} (hence a termination proof with the context processor would not be possible). Instead we linearize the term obtaining $x : (y : inf(s(z)))$ which we can use as canonical forbidden pattern. Indeed, $\mathcal{R}$ is Π-terminating when choosing $\Pi = \langle x : y : inf(s(z)), 2.2, h\rangle$.

As an alternative to the on-the-fly generation of forbidden patterns during the termination analysis with SCP_n processors, in some cases an (iterated) two phase process might be more efficient. There, Stage 4 of the above algorithm scheme is not carried out, i.e. no CDPs are deleted after the generation of forbidden patterns. Instead, in phase 2, the termination analysis starts from scratch using the generated forbidden patterns. If it fails, new forbidden patterns are generated and termination is analyzed again afterwards. This sequence of (separated) generation of forbidden patterns and termination analysis continues until termination is proved.

While at first glance the two phase approach seems to be less efficient than the on-the-fly generation of forbidden patterns during the termination analysis, it has an important advantage. In the phase of the generation of forbidden patterns an arbitrary subset of CDPs can be used for the synthesis of forbidden patterns. Since termination is proved separately, this does not affect the soundness of the approach. The concrete advantages of this approach are the following.

- For the termination analysis one is not restricted to the CDP framework. One can for instance use the transformation $\mathbb{T}$ of Section 4.4.1.
- When disregarding structural dependency pairs during the synthesis of forbidden patterns, the generated patterns are more intuitive, simpler and often suffice to obtain termination.
- When using the CDP framework, the generated (stable) forbidden patterns can be used to compute the concrete set of CDPs.
- The generation of forbidden patterns is more fine-grained, since not all sequences of CDPs of a given length are considered in the SCP_n processor, but only those contained in the specified subset of CDPs (which could for instance be specified by a human in a semi-automatic synthesis process). This results in fewer created forbidden patterns that might still be sufficient to yield Π-termination.

Example 70. *In Example 69 exactly the only non-structural dependency pair is used. Using the two phase approach the according forbidden pattern is found fully automatically.*

Chapter 5

VMTL - Vienna Modular Termination Laboratory

5.1 Introduction

During the last decade, remarkable progress has been made in the field of termination analysis of term rewriting systems. Despite termination being an undecidable property of TRSs, increasingly sophisticated methods have been developed to prove it for given systems. From these efforts several tools have emerged that are capable of proving termination (semi) automatically (cf. e.g. [31], [53], [3]).

The currently most powerful tools implement the dependency pair framework of [33] based on the idea of dependency pair analysis of [9]. This approach has at least two advantages. First, it seems to be the most powerful one, which is indicated by the latest results of a yearly termination competition ([1]). Second, the pure dependency pair framework is strictly modular, which means that concrete (correct) methods to prove termination within this framework can be combined, added and removed arbitrarily, without affecting the correctness of the tool. Thus, it is easy to extend tools implementing this framework by new methods (called *dependency pair processors* in the terminology of [33]).

VMTL (cf. `http://www.logic.at/vmtl/`) is a new termination tool that implements the dependency pair framework and focuses on openness, modularity and extensibility. It is easily extensible by new dependency pair processors, while providing the main technical infrastructure of termination tools such as thread and processor scheduling, timeout handling, input parsing, output formatting etc. In addition, VMTL contains implementations of several well-known methods of proving termination within the dependency pair framework. These include

- a reduction pair processor based on recursive path orderings,
- a reduction pair processor based on polynomial interpretations,
- narrowing and instantiation processors (see below for more details), and

- a size-change principle processor.

The two reduction pair processors are implemented via reduction to satisfiability problems and utilizing external SAT solvers, as it is state of the art at the time of writing. Benchmarks of VMTL on the set of TRSs used in the latest termination competition can be found below.

Another focus during the development of VMTL was applicability to (and suitability for) conditional term rewriting systems (CTRSs). Termination of such CTRSs is usually verified by transforming them into ordinary TRSs and deriving termination of the CTRSs from termination of the transformed TRSs. VMTL provides a public interface that allows users to plug in such transformations. Moreover, it includes the transformation of deterministic CTRSs in to context-sensitive TRSs from Chapter 3.

VMTL is also capable of proving termination of context-sensitive term rewriting systems (CSRSs). For this task the refined context-sensitive dependency pair approach of [2] is used. Using context-sensitive dependency pairs, and their property that they coincide with context-free dependency pairs for context-free term rewriting systems, allows VMTL to treat every TRS as CSRS, having a trivial replacement map in case of a context-free TRS.

5.2 User Interface

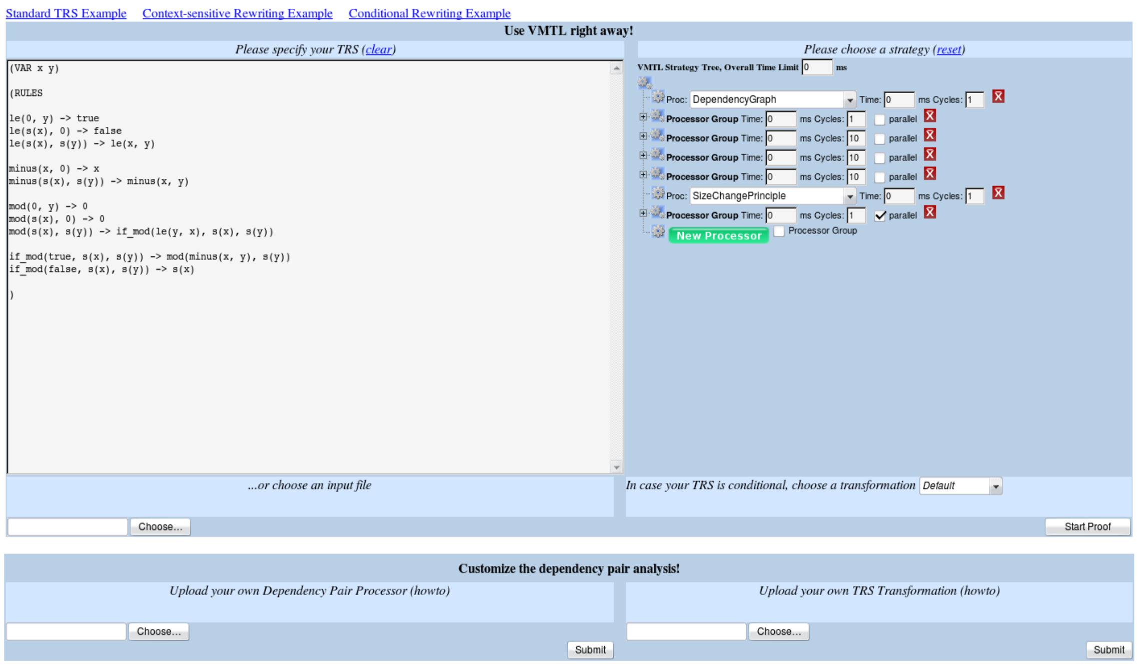
VMTL provides a command line interface for batch execution and a web interface that eases configuration and extension. Figure 5.1 shows a screenshot of the web interface right after startup. It contains fields for entering a TRS or alternatively uploading a file. VMTL exclusively accepts the input format specified for the termination competition. On the right-hand side the user can define the strategy, i.e., the order in which processors are applied, together with time constraints to be satisfied. At the bottom there are two fields allowing the user to upload new customized dependency pair processors and transformations, respectively.

5.2.1 User Defined Strategies

Since dependency pair processors often modify and simplify dependency pair problems, it is crucial to apply them in a reasonable order. Moreover, some processors are more time consuming than others. Thus, in order to achieve the goal of proving termination as quickly as possible, it is important to have a good strategy for processor application. VMTL allows the user to fully configure this strategy. It provides the following degrees of freedom in its specification.

- Dependency pair processors can be arbitrarily ordered.

- Time limits can be imposed on dependency pair processors.



Standard TRS Example Context-sensitive Rewriting Example Conditional Rewriting Example

Use VMTL right away!

Please specify your TRS (clear)

```
(VAR x y)

(RULES

le(0, y) -> true
le(s(x), 0) -> false
le(s(x), s(y)) -> le(x, y)

minus(x, 0) -> x
minus(s(x), s(y)) -> minus(x, y)

mod(0, y) -> 0
mod(s(x), 0) -> 0
mod(s(x), s(y)) -> if_mod(le(y, x), s(x), s(y))

if_mod(true, s(x), s(y)) -> mod(minus(x, y), s(y))
if_mod(false, s(x), s(y)) -> s(x)

)
```

Please choose a strategy (reset)

VMTL Strategy Tree, Overall Time Limit 0 **ms**

Proc: DependencyGraph Time: 0 ms Cycles: 1
Processor Group Time: 0 ms Cycles: 1 parallel
Processor Group Time: 0 ms Cycles: 10 parallel
Processor Group Time: 0 ms Cycles: 10 parallel
Processor Group Time: 0 ms Cycles: 10 parallel
Proc: SizeChangePrinciple Time: 0 ms Cycles: 1
Processor Group Time: 0 ms Cycles: 1 [✓ parallel]

...or choose an input file

In case your TRS is conditional, choose a transformation Default

Customize the dependency pair analysis!

Upload your own Dependency Pair Processor (howto)

Upload your own TRS Transformation (howto)

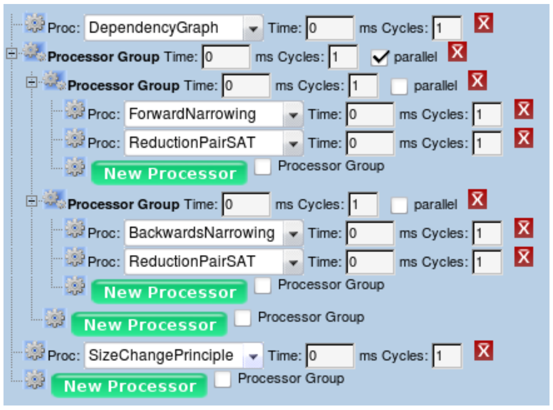
Figure 5.1: Snapshot of the VMTL web interface after start up.

- Dependency pair processors can be executed repeatedly (which can be helpful for instance for *narrowing* processors).
- Dependency pair processors can be hierarchically grouped to an arbitrary depth. Each group can be given a time limit. In case of contradicting time limits the shortest is used.
- Execution of dependency pair processors can be parallelized.

The semantics of parallelization in VMTL is that if one of the parallel branches finishes the termination proof, the whole execution is stopped immediately and the proof is presented to the user. Otherwise, if all parallel branches fail to prove termination, the termination proof continues according to the strategy with the dependency pair problems derived *before* the start of the parallel execution (cf. Example 1 below).

Graphically, a strategy is represented in VMTL as a tree, where each node can either be a dependency pair processor, or a "Group node". Group nodes are used to build groups of processors or other groups.

Example 1. Consider the snapshot of a strategy specification given in Figure 5.2. The basic execution order is from top to bottom and a time limit of zero means no time limit. The given strategy tree has the following meaning. First, the dependency graph processor (DependencyGraph) is applied. Then the execution of a processor group starts and the immediate child processors of this group are executed in parallel.



```
Proc: DependencyGraph Time: 0 ms Cycles: 1
Processor Group Time: 0 ms Cycles: 1 ✓ parallel
  Processor Group Time: 0 ms Cycles: 1 ☐ parallel
    Proc: ForwardNarrowing Time: 0 ms Cycles: 1
    Proc: ReductionPairSAT Time: 0 ms Cycles: 1
    ☐ Processor Group
  Processor Group Time: 0 ms Cycles: 1 ☐ parallel
    Proc: BackwardsNarrowing Time: 0 ms Cycles: 1
    Proc: ReductionPairSAT Time: 0 ms Cycles: 1
    ☐ Processor Group
  ☐ Processor Group
Proc: SizeChangePrinciple Time: 0 ms Cycles: 1
☐ Processor Group
```

Figure 5.2: Snapshot of the specification of a proof strategy in VMTL.

The child processors are again two groups, one sequentially executing the forward narrowing processor (ForwardNarrowing) followed by the reduction pair processor (ReductionPairSAT), the other one executing the backward narrowing processor (BackwardsNarrowing) followed by the reduction pair processor. If one of these groups is able to prove termination of the given (sets of) dependency pair problems, then the execution stops and the proof is displayed. Otherwise, the execution continues with the application of the size-change principle processor (SizeChangePrinciple) taking the output of the dependency graph processor (which was the last processor before the parallel group) as input.

5.3 VMTL API

VMTL provides a public interface that allows a user to easily build extensions. There are three basic functionalities that can be extended.

- New dependency pair processors can be added.
- New transformations, from (conditional/context-sensitive) TRSs to (context-sensitive) TRSs, can be added.
- New plug-ins for output formatting can be added.

5.3.1 Adding New Dependency Pair Processors

VMTL provides two interfaces *DPProcessor* and *ContextSensitiveDpProcessor* from which one has to be implemented by the user depending on whether the processor takes context-restrictions into account or not. In case DPProcessor is implemented, VMTL will make sure that the processor is not applied to context-sensitive dependency pair problems (even if the processor occurs in the strategy). Note that each context-sensitive DP processor is trivially a context-free one (as context-free DP problems can be seen as special cases of context-sensitive ones), thus ContextSensitiveDpProcessor is a “subinterface” (in an object oriented sense) of DPProcessor. See e.g. [2] for a justification that context-free dependency pair processors may in general not be applied to context-sensitive dependency pair problems.

A custom processor is given a dependency pair problem (that has possibly been processed by other processors before) and a set of (processor) parameters from VMTL and is required to return a set of derived dependency pair problems. In VMTL, the datastructure of a dependency pair problem consists of a set of dependency pairs, a rewrite system (possible context-restrictions are derivable from both through marked forbidden positions in terms) and a subsignature (cf. Section 3.2.4).

5.3.2 Adding New Transformations

Adding transformations to VMTL can be accomplished by implementing the interface *TrsToTrsTransformation*. Transformations in VMTL are not restricted to ones that transform conditional systems. The interface can be used to perform arbitrary preprocessing steps, such as semantic labelling etc. (this is the reason why the interface is not called *CtrsToTrsTransformation*). However, in case termination of a conditional rewrite system is to be proved, a transformation is mandatory. If none is specified by the user, VMTL will use the context-sensitive unraveling transformation of Chapter 3.

5.3.3 Customizing Output Formatting

The proof information accumulated by VMTL and the used dependency pair processors is represented in a simple native markup language, providing basic structuring and formatting tags. From this intermediate representation the actual (human or machine) readable output is created by so called *OutputWriter* objects. Out of the box, VMTL only supports HTML Output. However, the user may extend VMTL by additional OutputWriters through the *OutputWriter* interface and can thereby add additional support for proof certification (see e.g. [51]).

5.4 Implementation Details and Benchmarks

VMTL is entirely written in Java. The core module (i.e., without user interface) contains approximately 11.500 lines of code. MiniSat ([22]) is used as SAT solver.

We provide benchmarks for termination analysis of standard TRS and context-sensitive TRSs. Table 5.1 shows the performance of VMTL on the set of standard TRSs from the TPDB. Table 5.2 shows the benchmarks on the set of context-sensitive examples. Moreover, in Table 5.3 we compare the performance of VMTL with AProVE (Version 1.2) on a set of 24 CTRSs (newer versions of AProVE as well as other termination tools did not support proper deterministic conditional

Tool	**Successful Proofs**	**Number of Systems**
VMTL	588(105)	1391
AProVE	1226(231)	1391
TTT2	970(178)	1391
Jambox	810(60)	1391

Table 5.1: Benchmarks on standard TRSs from the TPDB (with a time limit of 60 seconds). The number in parenthesis shows how many of the successful proofs were actually disproofs.

Tool	**Successful Proofs**	**Number of Systems**
VMTL	65(0)	109
AProVE	94(0)	109

Table 5.2: Benchmarks on context-sensitive TRSs from the TPDB (with a time limit of 60 seconds). The number in parenthesis shows how many of the successful proofs were actually disproofs.

rewrite systems in our experiments). These examples were taken from the TPDB as well as [64], [73] and [34]. For several examples used in these benchmarks, proving termination depends on the methods described in Section 3.2.4.

Note that apart from the restricted set of proof methods available in VMTL, the inferior performance for standard and context-sensitive TRSs is due to the strict modularity. Strategies in VMTL cannot be history aware, which for instance prevents methods like *safe narrowing* from [33]. In addition, the proof strategy cannot (easily) be adapted to certain classes of TRSs such as applicative ones.

Tool	**Successful Proofs**	**Number of Systems**
VMTL	19(3)	24
AProVE	14(2)	24

Table 5.3: Benchmarks on conditional TRSs (with a time limit of 120 seconds). The number in parenthesis shows how many of the successful proofs were actually disproofs.

Chapter 6

Conclusion

6.1 Summary

In Chapter 3 the successful application of context-sensitive rewriting to the problem of simulating conditional rewriting by unconditional rewriting was illustrated. We were able to prove that by considering a context-sensitive variant of a simple and well-known encoding of (deterministic) CTRSs simulation soundness and simulation completeness are guaranteed. This is to say that the transformed system allows for exactly those reductions from old terms to old terms that are possible in the conditional one itself.

Recently, it has been shown that the same transformation without context-sensitivity is still simulation-sound and simulation-complete for certain classes of CTRSs (cf. [35] for details). However, these classes are rather restricted in that extra variables in conditions of right-hand sides of (conditional) rewrite rules may not occur.

One potential application for a simulation-sound transformation from DCTRSs into unconditional rewrite systems is the verification of operational termination of DCTRSs by verifying termination of the corresponding transformed systems. To this end we introduced the notion of context-sensitive quasi-reductivity, which is a context-sensitive variant of quasi-reductivity and (in contrast to quasi reductivity) proved to be equivalent to operational termination for DCTRSs. Context-sensitive quasi-reductivity of a DCTRS in turn is shown to be equivalent to context-sensitive termination of the corresponding unraveled system on terms over the original signature of the DCTRS, thus yielding a characterization of operational termination of DCTRSs by a local termination property of the corresponding unraveled unconditional CSRSs.

Complementing this result, we proved that $C_{\mathcal{E}}$-operational termination of a DCTRS is equivalent to $C_{\mathcal{E}}$-termination of the corresponding transformed CTRS. $C_{\mathcal{E}}$-termination is an important generalization of termination that is "more modular" than ordinary termination and thus preferable in proofs of termination based on divide-and-conquer strategies. For instance, $C_{\mathcal{E}}$-termination is modular for disjoint unions but also - more interestingly - for a certain practically relevant class of hierarchical combinations of rewrite systems called proper extensions ([73][Chapter 8]). These proper extensions are practically more relevant than combinations with disjoint signatures, since they can be understood as programs that are built upon other programs (libraries) in a hierarchical way while still many important modularity results hold.

In order to practically exploit the relaxed (local) termination condition of an unraveled CSRS that suffices to imply operational termination of the corresponding DCTRS, we provided a slight extension of the well-known dependency pair framework that takes subsignatures corresponding to local termination properties into account when dealing with CSRSs obtained by unravelings from DCTRSs. Building upon this extended DP framework we proposed narrowing and instantiation DP processors that are able to prove local termination of termination problems that are not globally terminating.

In Chapter 4 we introduced the notion of rewriting with forbidden patterns which is a proper generalization of context-sensitive rewriting utilized in Chapter 3. The basic idea is that replacements should be allowed or forbidden depending on the concrete shape of a redex and the context it is situated in. This generalizes context-sensitive rewriting in two dimensions. First, restrictions based on contexts in which redexes appear can be more complex than having allowed and forbidden arguments of functions. Second, the shape of the redex itself, i.e. the corresponding substitution, can be used to specify its status. Moreover, the forbidden entities in rewriting with forbidden patterns are subject to specification and not fixed. That is to say that in a forbidden pattern it can be specified whether a whole subterm, a single position or a set of positions above a predetermined position are forbidden for reduction, whereas in context-sensitive rewriting arguments of functions and thus whole subterms are forbidden by definition.

We systematically developed an infrastructure for rewriting with forbidden patterns based on a series of results covering the areas of completeness w.r.t. the simulation of unrestricted rewriting, confluence criteria and automated termination analysis. Moreover, we tackled the problem of automatically synthesizing suitable forbidden patterns for a given rewrite system.

Our main result on completeness of rewriting with forbidden patterns in simulating unrestricted rewriting is based on the notion of canonical forbidden patterns. Canonical forbidden patterns have a certain shape and properties relative to TRSs they are used for. When performing rewriting with canonical forbidden patterns, forbidden pattern normal forms are head-normal forms w.r.t. unrestricted rewriting.

Regarding confluence we provided two criteria based on Newman's Lemma [68] and the diamond property [76] of rewrite systems. It turned out that substantial restrictions on the patterns have to be imposed in order to soundly apply these criteria.

Finally, we provided two approaches to automatically verify termination of rewriting with forbidden patterns. The first approach is based on the transformation $\mathbb{T}$, transforming rewrite systems with forbidden linear h-patterns into ordinary rewrite systems while preserving termination and non-termination on ground terms. The second approach handling general h- and b-patterns is based on a modified dependency pair framework taking into account contexts of function calls represented by dependency pairs. In this contextual dependency pair framework we defined two processors that analyze nested contexts of dependency pairs in order to identify chains that do not adhere to the forbidden pattern restrictions and, consequently, can be excluded from the termination analysis. The latter approach can in particular be used to prove outermost termination of TRSs, since outermost rewriting can be equivalently expressed by a set of associated b-patterns for any given rewrite system.

Finally, in Chapter 5 we introduced the termination tool VMTL, which incorporates implementations of all termination methods developed in this thesis as well as a set of standard methods for ordinary and context-sensitive rewriting. The tool features a web front end with user-definable proof search strategy. Moreover, public programming interfaces for the modular addition of processors in the context-sensitive dependency pair framework of [2] are available. VMTL has been participating in the yearly competition of termination tools for TRS since 2009.

6.2 Related Work

6.2.1 Transforming Conditional Rewrite Systems

In Section 3.4 we discussed transformations of CTRSs and gave pointers to the literature where context-sensitivity and/or other strategies have been used to obtain soundness w.r.t. simulations when using (variants of) unraveling transformations. In [35] it was shown that for normal 1-CTRSs several properties are sufficient to ensure simulation-soundness of the ordinary unraveling transformation of Definition 2. Among the more important of these properties of a DCTRS are

- weak left-linearity, i.e. each rule is either left-linear or unconditional and every non-linear variable of the lhs does not occur in the rhs, and
- non-erasingness, and
- confluence.

Hence, if analyzing conditional rewrite systems that belong to the class of normal 1-CTRSs and have either of these properties, the additional context-sensitivity of the unraveling of Definition 4 as compared to the one of Definition 2 is not needed in order to obtain simulation-soundness. This result is interesting, since in many cases the use of context-sensitivity entails additional complications in the handling and analysis of the unconditional systems obtained by the transformations. Thus, the above results enable us to use the more complicated unraveling of Definition 4 “on demand”, i.e. only in situations where the simpler non-context-sensitive unraveling does not have the desired properties (e.g. simulation soundness).

Other classes of transformations transforming conditional rewrite systems into unconditional ones are the ones of [8, 85, 83, 34]. There, in contrast to unravelings, the focus lies on simulating conditional rewriting in a practically efficient way (while unravelings are traditionally used to reason about properties of the CTRSs such as operational termination). The central property that these transformations are trying to achieve is *computational equivalence* ([83]). Informally, if a transformation has this property, then no backtracking is needed to compute normal forms w.r.t. the conditional system in the unconditional one. Unravelings do not have this property.

Example 71. *Consider the following conditional rewrite system $\mathcal{R}$ defining a function* **insert***, inserting a number into a sorted list of numbers.*

$$\begin{aligned}
0 \leq x &\to \top\\
s(x) \leq 0 &\to \bot\\
s(x) \leq s(y) &\to x \leq y\\
insert(x, nil) &\to x : nil\\
insert(x, y : ys) &\to x : y : ys \Leftarrow x \leq y \to^* \top\\
insert(x, y : ys) &\to y : insert(x, ys) \Leftarrow x \leq y \to^* \bot
\end{aligned}$$

The unraveled CSRS $U_{cs}(\mathcal{R})$ is obtained from $\mathcal{R}$ by replacing the conditional rules of $\mathcal{R}$ by

$$\begin{aligned}
insert(x, y : ys) &\to U_1^1(x \leq y, x, y, ys)\\
U_1^1(\top, x, y, ys) &\to x : y : ys\\
insert(x, y : ys) &\to U_1^2(x \leq y, x, y, ys)\\
U_1^2(\bot, x, y, ys) &\to y : insert(x, ys)
\end{aligned}$$

and using a replacement map μ given by $\mu(U_1^1) = \mu(U_1^2) = \{1\}$ and $\mu(f) = \{1, \ldots, ar(f)\}$ for all $f \notin \{U_1^1, U_1^2\}$. Now consider the following $U_{cs}(\mathcal{R})$ reduction

$$\begin{aligned}
insert(s(0), 0 : s(s(0)) : nil) &\to U_1^1(s(0) \leq 0, s(0), 0, s(s(0)) : nil)\\
&\to U_1^1(\bot, s(0), 0, s(s(0)) : nil)
\end{aligned}$$

The final term is a $U_{cs}(\mathcal{R})$-normal form. Hence, we cannot compute the expected result $0 : s(0) : s(s(0)) : nil$ from it. In order to obtain this result, we should have applied the rule $insert(x, y : ys) \to U_1^2(x \leq y, x, y, ys)$ instead of the rule $insert(x, y : ys) \to U_1^1(x \leq y, x, y, ys)$ in the first step of the above reduction sequence. Hence, in order to obtain the correct result in the above reduction sequence backtracking is needed.

When using transformations that produce TRSs that are computationally equivalent to the original CTRSs, such backtracking is not needed. Indeed, $\mathcal{R}$ can be transformed into a computationally equivalent TRS. The reason is that it is strongly deterministic ([73][Definition 7.2.35]), operationally terminating (which is easily *verified by the methods of Chapter 3 and VMTL) and confluent by [73][Theorem 7.3.2] (since all conditional critical pairs are infeasible [73][Definition 7.1.8] and thus joinable). Hence, the transformation of [34] yields a computationally equivalent TRS (by [34][Theorem 5]).*

6.2.2 Restrictions in Term Rewriting

Proper restrictions of term rewriting have been studied extensively in recent years. Ideas range from lazy rewriting [27, 79, 56], over context-sensitive rewriting [55, 60, 4, 3] and rewriting with strategy annotations [23, 59] to on-demand rewriting [58, 56] and rewriting with on-demand strategy annotations [7]. For a detailed survey and comparison of these approaches we refer to [77].

In particular, the approaches incorporating a notion of laziness going beyond the blocking of certain arguments of function symbols (as in context-sensitive rewriting), such as lazy rewriting, on-demand rewriting and rewriting with on-demand strategy annotations have proven to be hard to understand, handle and analyze. All these approaches rely on concepts going beyond matching and comparison of positions in terms that are integral to term rewriting as such. In the case of lazy rewriting function symbols (in terms) are labelled by flags indicating whether the terms rooted by them are to be evaluated eagerly or in a lazy fashion. On-demand rewriting relies on "partial matches" to determine the status of subterms and rewriting with on-demand strategy annotations explicitly keeps track of positions where rewriting takes place in terms and uses labels for function symbols as well.

Hence, context-sensitive rewriting has emerged from all these formalisms as the most tractable and practically most relevant one. At the time of writing this thesis the main focus in the research on context-sensitive rewriting is the automated verification of termination of context-sensitive rewrite systems [5, 41, 2, 42, 40]. There, in particular, the use of certain variants of context-sensitive dependency pairs and DP processors in the resulting DP frameworks are studied. Here, it is interesting that there are several notions of context-sensitive dependency pair framework (cf. [40]). In Section 3.2.4 of this thesis we use one particular framework, namely the one introduced in [2]. It is a natural question whether our results there carry over also to other frameworks. Moreover, other frameworks might be even more suitable to analyze termination of CSRSs obtained from DCTRSs by context-sensitive unravelings. Moreover, the ideas on which these other, more general dependency pair frameworks are based on may be generalized and reused to also improve the forbidden pattern dependency pair framework introduced in this thesis.

Summarizing, we believe that the widespread interest in termination methods of context-sensitive rewriting indicates that termination is a (maybe the) central aspect that one wants to achieve when using proper restrictions of rewriting.

6.3 Discussion

Term rewriting is an important theoretical tool in many fields of computer science such as formal methods, automated deduction and programming languages. A particularly interesting and practically promising application of rewriting is the use of rewriting logic ([65, 14, 66]) as a mathematical model of the formal specification and programming language Maude ([15]). Having a precise formal semantics of a programming language has many advantages and makes formal reasoning over and verification of programs possible where it would have been impossible or considerably harder if languages without a well-defined formal semantics had been used.

The Maude specification and programming language and rewriting logic make use of results of term rewriting in several ways. First, an important part of Maude programs are equations. However, since equational reasoning in Maude is carried out in a directed way based on term rewriting, this reasoning is only complete if the oriented equations (that thus form a rewrite system) are confluent. Moreover, in this case termination of this rewriting system implies termination of equational reasoning. Hence, there is a great interest in the automated verification of these properties for Maude programs that resulted in the development of the Maude Church-Rosser Checker (CRC [21]) and the Maude Termination Tool (MTT [19]) as parts of the Maude formal tool environment ([16]). These tools make heavy use of available results for the automated confluence and termination analysis of TRSs.

Second, the Maude engine executes given specifications by term rewriting. That is to say that reasoning in the rewrite logic theories represented by Maude programs is performed by rewriting. Hence, inevitably reduction strategies and proper restrictions of rewriting play an important role. For instance, Maude allows for explicit definitions of operator evaluation strategies, also including laziness.

The reason we picked Maude to briefly describe a practical application of rewriting in the area of formal methods resp. programming languages, is that it nicely illustrates two areas of work in term rewriting also present in this thesis. First, there is the verification of properties of given TRSs and second there are strategies used to efficiently compute normal forms w.r.t. a given TRS. In this sense the two main Chapters (Chapters 3 and 4) complement each other in that they illustrate the use of proper restrictions of rewriting in the analysis of operational termination of DCTRSs (Chapter 3) and in defining evaluation strategies that have desirable practical properties (Chapter 4).

Another important duality in these two main chapters of the thesis is that they represent respective ends on the scale of scientific maturity. The field of conditional rewrite systems dealt with in Chapter 3 is comparatively old (within the field of term rewriting as a whole), well-studied and understood. Our work revisits the field and turns the attention to context-sensitivity and the implications of its use in transformations from DCTRSs into unconditional TRS. Some of the main questions that arise when doing this are answered. Our results include soundness and completeness w.r.t. simulations of context-sensitive unravelings, the characterization of operational termination of DCTRSs by a local termination property of CSRSs and the theoretically interesting observation that quasi-reductivity becomes equivalent to quasi-decreasingness when additionally allowing for context-sensitivity in the used signature extensions and considering only substitutions ranging into the original terms in the ordering constraints (Corollary 4).

The main theorems of this chapter characterize the properties that we are interested in (e.g. operational termination and $C_{\mathcal{E}}$-operational termination) in terms of properties of unconditional rewrite systems. We claim that these characterizations leave little to no room for improvement regarding the encoding of operational termination by means of unraveling transformations (while the question of analyzing local termination of course leaves much space for future research).

In contrast to that, Chapter 4 must be understood as a starting point and formal basis for many areas where rewriting with forbidden patterns could be applied successfully. While a principal infrastructure for rewriting with forbidden patterns is provided (including criteria for confluence, termination and completeness), there are plenty of opportunities for improving and extending our results as well as the formalism itself. Indeed, we hope that the definitorial simplicity and great power and flexibility of rewriting with forbidden patterns will lead to flourishing interest in and applications of this approach.

Bibliography

[1] The termination competition. `http://termcomp.uibk.ac.at/`.

[2] B. Alarcón, F. Emmes, C. Fuhs, J. Giesl, R. Gutiérrez, S. Lucas, P. Schneider-Kamp, and R. Thiemann. Improving context-sensitive dependency pairs. In *Proceedings of the 15th International Conference on Logic for Programming, Artificial Intelligence, and Reasoning (LPAR'08)*, pages 636–651. Springer-Verlag, 2008.

[3] B. Alarcón, R. Gutiérrez, J. Iborra, and S. Lucas. Proving termination of context-sensitive rewriting with MU-TERM. In *Proceedings of the 6th Spanish Conference on Programming and Computer Languages (PROLE'2006) - Selected papers*, volume 188 of *Electronic Notes in Theoretical Computer Science (ENTCS)*, pages 105–115. Elsevier, 2007.

[4] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-sensitive dependency pairs. In s. Arun-Kumar and N. Garg, editors, *Proceedings of the 26th International Conference on Foundations of Software Technology and Theoretical Computer Science, FST&TCS'06*, volume 4337 of *Lecture Notes in Computer Science (LNCS)*, pages 297–308. Springer, 2006.

[5] B. Alarcón, R. Gutiérrez, and S. Lucas. Context-sensitive dependency pairs. *Information and Computation*, 208(8):922 – 968, 2010.

[6] B. Alarcón, S. Lucas, and J. Meseguer. A dependency pairs framework for AvC termination. In P. Ölveczky, editor, *Proceedings of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10)*, 2010.

[7] M. Alpuente, S. Escobar, B. Gramlich, and S. Lucas. On-demand strategy annotations revisited: An improved on-demand evaluation strategy. *Theoretical Computer Science*, 411(2):504–541, 2010.

[8] S. Antoy, B. Brassel, and M. Hanus. Conditional narrowing without conditions. In *Proceedings of the 5th ACM SIGPLAN International Conference on Principles and Practice of Declaritive Programming (PPDP'03)*, pages 20–31. ACM, 2003.

[9] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[10] J. Avenhaus and C. Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In F. Pfenning, editor, *Proceedings of the 5th International Conference on Logic Programming and Automated Deduction (LPAR'94)*, volume 822 of *Lecture Notes in Artificial Intelligence*, pages 215–229. Springer, 1994.

[11] F. Baader and T. Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[12] J. A. Bergstra and J. W. Klop. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences*, 32:323–362, 1986.

[13] C. Borralleras, S. Lucas, and A. Rubio. Recursive path orderings can be context-sensitive. In A. Voronkov, editor, *Proceedings of 18th International Conference on Automated Deduction (CADE'02)*, volume 2392 of *Lecture Notes in Computer Science (LNCS)*, pages 314–331. Springer, 2002.

[14] A. Bouhoula, J.-P. Jouannaud, and J. Meseguer. Specification and proof in membership equational logic. *Theoretical Computer Science*, 236:35–132, 2000.

[15] M. Clavel, F. Durán, S. Eker, P. Lincoln, N. Martí-Oliet, J. Meseguer, and J. Quesada. *Maude: Specification and Programming in Rewriting Logic.* SRI International, 1999.

[16] M. Clavel, F. Durán, J. Hendrix, S. Lucas, J. Meseguer, and P. Ölveczky. The maude formal tool environment. In *Proceedings of the 2nd International Conference on Algebra and Coalgebra in Computer Science (CALCO'07)*, volume 4624 of *Lecture Notes in Computer Science (LNCS)*, pages 173–178. Springer-Verlag, 2007.

[17] H. Comon, M. Dauchet, R. Gilleron, C. Löding, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: `http://www.grappa.univ-lille3.fr/tata`, 2007. release October, 12th 2007.

[18] F. Durán, S. Lucas, C. Marché, J. Meseguer, and X. Urbain. Proving operational termination of membership equational programs. *Higher-Order and Symbolic Computations*, 21(1-2):59–88, 2008.

[19] F. Durán, S. Lucas, and J. Meseguer. MTT: The maude termination tool (system description). In A. Armando, P. Baumgartner, and G. Dowek, editors, *Proceedings of the 4th International Joint Conference on Automated Reasoning (IJCAR'08)*, volume 5195 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 313–319. Springer-Verlag, 2008.

[20] F. Durán, S. Lucas, J. Meseguer, C. Marché, and X. Urbain. Proving termination of membership equational programs. In *Proceedings of the 2004 ACM SIGPLAN symposium on Partial evaluation and semantics-based program manipulation (PEPM'04)*, pages 147–158. ACM, 2004.

[21] F. Durán and J. Meseguer. A Church-Rosser checker tool for conditional order-sorted equational maude specifications. In P. Ölvecky, editor, *Proceedings of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10)*, 2010.

[22] N. Eén and N. Sörensson. An extensible SAT-solver. In G. Goos, J. Hartmanis, and J. van Leeuwen, editors, *Theory and Applications of Satisfiability Testing*, volume 2919 of *Lecture Notes in Computer Science (LNCS)*, pages 333–336. Springer, 2004.

[23] S. Eker. Term rewriting with operator evaluation strategy. In *In 2nd International Workshop on Rewriting Logic and its Applications (WRLA'98)*, volume 15 of *Electronic Notes in Theoretical Computer Science*. Elsevier, 1998.

[24] J. Endrullis. Jambox. Available at `http://joerg.enrullis.de/`.

[25] J. Endrullis and D. Hendriks. From outermost to context-sensitive rewriting. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science (LNCS)*, pages 305–319. Springer, 2009.

[26] J. Endrullis, R. Vrijer, and J. Waldmann. Local termination. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science (LNCS)*, pages 270–284. Springer, 2009.

[27] W. Fokkink, J. Kamperman, and P. Walters. Lazy rewriting on eager machinery. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 22(1):45–86, 2000.

[28] D. P. Friedman and D. S. Wise. CONS should not evaluate its arguments. In I. S. Michaelson and R. Milner, editors, *Proceedings of the 3rd Colloquium on Automata, Languages and Programming*, pages 257–284. Edinburgh University Press, 1976.

[29] H. Ganzinger and U. Waldmann. Termination proofs of well-moded logic programs via conditional rewrite systems. In M. Rusinowitch and J.-L. Rémy, editors, *Proceedings of the 3rd International Workshop on Conditional Term Rewriting Systems (CTRS'92)*, volume 656 of *Lecture Notes in Computer Science (LNCS)*, pages 430–437. Springer, 1993.

[30] J. Giesl and A. Middeldorp. Transformation techniques for context-sensitive rewrite systems. *Journal of Functional Programming*, 14(4):379–427, 2004.

[31] J. Giesl, P. Schneider-Kamp, and R. Thiemann. Aprove 1.2: Automatic termination proofs in the dependency pair framework. In U. Furbach and N. Shankar, editors, *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Computer Science (LNCS)*, pages 281–286. Springer, 2006.

[32] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In B. Gramlich, editor, *Proceedings of the 5th International Workshop on Frontiers of Combining Systems, FROCOS'05*, volume 3717 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 216–231. Springer, 2005.

[33] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[34] K. Gmeiner and B. Gramlich. Transformations of conditional rewrite systems revisited. In A. Corradini and U. Montanari, editors, *Recent Trends in Algebraic Development Techniques (WADT'08) – Selected Papers*, volume 5486 of *Lecture Notes in Computer Science*, pages 166–186. Springer, 2009.

[35] K. Gmeiner, B. Gramlich, and F. Schernhammer. On (un)soundness of unravelings. In C. Lynch, editor, *Proceedings of the 21st International Conference on Rewriting Techniques and Applications (RTA'10)*, volume 6 of *LIPIcs (Leibniz International Proceedings in Informatics)*, pages 119–134, 2010.

[36] B. Gramlich and S. Lucas. Modular termination of context-sensitive rewriting. In *Proceedings of the 4th ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'02)*, pages 50–61. ACM, 2002.

[37] B. Gramlich and F. Schernhammer. Extending context-sensitivity in term rewriting. In M. Fernandez, editor, *Proceedings of the 9th International Workshop on Reduction Strategies in Rewriting and Programming (WRS'09)*, volume 15 of *Electronic Proceedings in Theoretical Computer Science (EPTCS)*, pages 56–68, 2009.

[38] B. Gramlich and F. Schernhammer. Outermost termination via contextual dependency pairs. In P. Schneider-Kamp, editor, *Proceedings of the 11th International Workshop on Termination (WST'10)*, 2010.

[39] B. Gramlich and F. Schernhammer. Termination of rewriting with - and automated synthesis of - forbidden patterns. In H. Kirchner and C. Muñoz, editors,

Prelim. Proceedings of the 1st International Workshop on Strategies in Rewriting, Proving and Programming (IWS'10), pages 13–17, 2010. Full version to appear in the Final Proceedings of IWS'10, EPTCS, 2010.

[40] R. Gutiérrez. *Automatic Proofs of Termination of Context-Sensitive Rewriting.* PhD thesis, Departamento de Sistemas Informàticos i Computación, Universidad Politècnica de Valencia, Valencia, Spain, 2010.

[41] R. Gutiérrez and S. Lucas. Proving termination in the context-sensitive dependency pair framework. In *Proceedings of the 8th International Workshop on Rewriting Logic and its Applications (WRLA'10)*, volume 6381 of *Lecture Notes in Computer Science (LNCS)*, pages 19–35, 2010.

[42] R. Gutiérrez, S. Lucas, and X. Urbain. Usable rules for context-sensitive rewrite systems. In A. Voronkov, editor, *Proceedings of the 19th International Conference on Rewriting Techniques and Applications (RTA '08)*, volume 5117 of *Lecture Notes in Computer Science (LNCS)*, pages 126–141. Springer-Verlag, 2008.

[43] M. Hanus. The integration of functions into logic programmming. *Journal of Logic Programming*, 19/20:583–628, 1994.

[44] P. Henderson and J. H. Morris Jr. A lazy evaluator. In *Proceedings of the 3rd ACM SIGACT-SIGPLAN Symposium on Principles on Programming Languages(POPL'76)*, pages 95–103. ACM, 1976.

[45] G. Huet. Confluent reductions: abstract properties and applications to term rewriting systems. *Journal of the ACM*, 24(4):797–821, 1980.

[46] G. Huet and J.-J. Lévy. Computations in orthogonal rewriting systems. *Computational Logic, Essays in Honor of Alan Robinson (eds. Jean-Louis Lassez and Gordon Plotkin)*, pages 396–443, 1991. Previous version: *Call by Need Computations in Non-Ambigious Linear Term Rewriting Systems*, report 359, INRIA, 1979.

[47] S. P. Jones, editor. *Haskell 98 Language and Libraries: The revised report.* Cambridge University Press, 2003.

[48] S. Kaplan. Conditional rewrite rules. *Theoretical Computer Science*, 33(2-3):175–193, 1984.

[49] R. Kennaway. A conflict between call-by-need computation and parallelism. In N. Derschowitz and N. Lindenstrauss, editors, *Proceedings of the 4th International Workshop on Conditional and Typed Rewriting Systems (CTRS'94)*, volume 968 of *Lecture Notes in Computer Science (LNCS)*, pages 247–261. Springer, 1995.

[50] J. W. Klop and A. Middeldorp. Sequentiality in orthogonal term rewriting systems. *Journal of Symbolic Computation*, 12:161–195, 1991.

[51] A. Koprowski. *Termination of Rewriting and Its Certification.* PhD thesis, Eindhoven University of Technology, 2008.

[52] M. Korp and A. Middeldorp. Beyond dependency graphs. In *Proceedings of the 22nd International Conference on Automated Deduction, CADE'09*, volume 5663 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 339–354. Springer, 2009.

[53] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean termination tool 2. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science (LNCS)*, pages 295–304. Springer, 2009.

[54] A. Leitsch. *The resolution calculus.* EATCS Texts in Theoretical Computer Science, Springer, 1997.

[55] S. Lucas. Context-sensitive computations in functional and functional logic programs. *Journal of Functional and Logic Programming*, 1998(1):1–61, 1998.

[56] S. Lucas. Context-sensitive rewriting, lazy rewriting, and on-demand rewriting. In M. Hanus, editor, *Proc. of International Workshop on Functional and (Constraint) Logic Programming (WFLP'01)*, volume 2017 of *Technical Report*, pages 197–210. Christian-Albrechts-Universitaet zu Kiel, September 2001.

[57] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In *Proceedings of the 3rd ACM SIGPLAN International Conference on Principles and Practice of Declarative Programming (PPDP'01)*, pages 82–93, 2001.

[58] S. Lucas. Termination of on-demand rewriting and termination of OBJ programs. In H. Sondergaard, editor, *Proc. of 3rd International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming (PPDP'01)*, pages 82–93. ACM Press, New York, 2001.

[59] S. Lucas. Termination of rewriting with strategy annotations. In A. Voronkov and R. Nieuwenhuis, editors, *Proc. of 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR'01*, volume 2250 of *Lecture Notes in Artificial Intelligence*, pages 669–684, La Habana, Cuba, December 2001. Springer-Verlag, Berlin.

[60] S. Lucas. Context-sensitive rewriting strategies. *Information and Computation*, 178(1):294–343, 2002.

[61] S. Lucas, C. Marché, and J. Meseguer. Operational termination of conditional term rewriting systems. *Information Processing Letters*, 95(4):446–453, 2005.

[62] S. Lucas and J. Meseguer. Order-sorted dependency pairs. In *Proceedings of the 10th international ACM SIGPLAN conference on Principles and practice of declarative programming (PPDP'08)*, pages 108–119, New York, NY, USA, 2008. ACM.

[63] M. Marchiori. Unravelings and ultra-properties. Technical Report 8, University of Padova, Italy, 1995. 37 pages, long version of [64].

[64] M. Marchiori. Unravelings and ultra-properties. In M. Hanus and M. M. Rodríguez-Artalejo, editors, *Proceedings of the 5th International Conference on Algebraic and Logic Programming (ALP'96)*, volume 1139 of *Lecture Notes in Computer Science (LNCS)*, pages 107–121. Springer, 1996.

[65] N. Martí-Oliet and J. Meseguer. Rewriting logic as a logical and semantic framework. Technical report, SRI International, 1993.

[66] J. Meseguer. Software specification and verification in rewriting logic. *Models, Algebras and Logic of Engineering Software*, 191:133–194, 2003.

[67] R. Milner, M. Tofte, and D. M. Queen. *The Definition of Standard ML.* MIT Press, Cambridge, MA, USA, 1997.

[68] M. H. A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.

[69] N. Nishida, M. Sakai, , and T. Sakabe. On simulation-completeness of unraveling for conditional term rewriting systems. In *LA Symposium 2004-7 Summer*, pages 7–1–7–6, 2004.

[70] N. Nishida, M. Sakai, and T. Sakabe. Partial inversion of constructor term rewriting systems. In J. Giesl, editor, *Proceedings of the 16th International Conference on Term Rewriting and Applications (RTA'2005)*, volume 3467 of *Lecture Notes in Computer Science (LNCS)*, pages 264–278. Springer, 2005.

[71] M. J. O'Donnell. *Computing in systems described by equations.* Springer-Verlag New York, Inc., 1977.

[72] E. Ohlebusch. Transforming conditional rewrite systems with extra variables into unconditional systems. In H. Ganzinger, D. A. McAllester, and A. Voronkov, editors, *Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Artificial Intelligence (LNAI)*, pages 111–130. Springer, 1999.

[73] E. Ohlebusch. *Advanced topics in term rewriting.* Springer, 2002.

[74] M. Oyamaguchi. NV-sequentiality: A decidable condition for call-by-need computations in term rewriting systems. *SIAM Journal on Computation*, 22(1):114–145, 1993.

[75] M. Raffelsieper and H. Zantema. A transformational approach to prove outermost termination automatically. In A. Middeldorp, editor, *Proceedings of the 8th International Workshop on Reduction Strategies*, volume 237 of *Electronic Notes in Theoretical Computer Science (ENTCS)*, pages 3–21. Elsevier, 2009.

[76] B. K. Rosen. Tree-manipulating systems and church-rosser theorems. In *Proceedings of the 2nd Annual ACM Symposium on Theory of Computing (STOC'70)*, pages 117–127. ACM, 1970.

[77] F. Schernhammer. On context-sensitivity in term rewriting. Master's thesis, Vienna University of Technology, 2007.

[78] F. Schernhammer and B. Gramlich. On proving and characterizing operational termination of deterministic conditional rewrite systems. In D. Hofbauer and A. Serebrenik, editors, *Informal Proceedings of the 9th International Workshop on Termination (WST'07)*, pages 82–85, 2007.

[79] F. Schernhammer and B. Gramlich. Termination of lazy rewriting revisited. In J. Giesl, editor, *Preliminary Proceedings of the 7th International Workshop on Reduction Strategies in Rewriting and Programming (WRS'07)*, pages 28–42, 2007. Full version in volume 204 of ENTCS (Final Proceedings of WRS'07), pages 35–51. Elsevier, 2008.

[80] F. Schernhammer and B. Gramlich. On operational termination of deterministic conditional rewrite systems. In T. Uustalu, J. Vain, and J. Ernits, editors, *Informal Proceedings of the 20th Nordic Workshop on Programming Theory (NWPT'08)*, pages 84–86, 2008.

[81] F. Schernhammer and B. Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. *Journal of Logic and Algebraic Programming*, 79(7):659–688, 2009. Revised selected papers of NWPT 2008, Tarmo Uustalu and Jüri Vain, editors.

[82] F. Schernhammer and B. Gramlich. VMTL – A modular termination laboratory. In R. Treinen, editor, *Proceedings of the 20th International Conference on Rewriting Techniques and Applications (RTA'09)*, volume 5595 of *Lecture Notes in Computer Science (LNCS)*, pages 285–294. Springer, 2009.

[83] T.-F. Şerbănuţă and G. Roşu. Computationally equivalent elimination of conditions. In F. Pfenning, editor, *Proceedings of the 17th International Conference on Rewriting Techniques and Applications (RTA'06)*, volume 4098 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2006.

[84] R. Thiemann. From outermost termination to innermost termination. In M. Nielsen, A. Kucera, P. B. Miltersen, C. Palamidessi, P. Tuma, and F. D. Valencia, editors, *Proceedings of the 35th Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM'09)*, volume 5404 of *Lecture Notes in Computer Science (LNCS)*, pages 533–545. Springer, 2009.

[85] P. Viry. Elimination of conditions. *Journal of Symbolic Computation*, 28(3):381–401, 1999.

List of Figures

List of Tables

Index

Curriculum Vitae

Felix Schernhammer

Address

Favoritenstraße 9/E185/2
A-1040 Vienna
Austria
Email: `felix.schernhammer@gmx.at`
Homepage: `www.logic.at/people/schernhammer/`

Personal Details

Date of birth: June 12, 1983
Nationality: Austrian

Education

10/2002–09/2005	Undergraduate studies in computer science at the Vienna University of Technology with a focus on software engineering, completed on September 7, 2005 with distinction.
10/2005–02/2007	Master studies in computer science at the Vienna University of Technology, completed on February 8, 2007 with distinction.
	Specialization: theoretical computer science, logics
	Thesis: *On context-sensitivity in term rewriting*; The thesis contains a comparative analysis of several approaches to extensions/restrictions in term rewriting and their interrelations. Supervisor: Prof. Dr. Bernhard Gramlich.
Since 03/2007	Ph.D. study at the Vienna University of Technology
	Project title: *Applications and Generalizations of Context-Sensitive Term Rewriting*; Supervisor: Prof. Dr. Bernhard Gramlich.
	Intended date of graduation: January 31, 2011.

PhD Thesis Topic

A novel approach to syntactic restrictions in term rewriting is proposed and analyzed. The basic idea is that so called *forbidden patterns* specify positions in a term that are forbidden for reduction. The basic goal is to develop criteria under which rewriting with this restriction is feasible in that useful results can be computed, while being restrictive enough to have desirable operational properties like termination.

Working Experience

07/2001–02/2002	Military service (mandatory in Austria).
05/2002–09/2002	Employed at a large Austrian pension fund (ÖPAG AG).
2003–2006	Employed part time at ÖPAG AG.
10/2004–02/2006	Employed as tutor for a basic course in theoretical computer science at the Vienna University of Technology.
07/2007–02/2008	Employed at Smart Information Systems GmbH.

Language Knowledge

German	**native**
English	**good**

Publications

All of my publications are available on my homepage at
`http://www.logic.at/people/schernhammer/`