FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Transformational Approaches for Conditional Term Rewrite Systems

PhD THESIS

submitted in partial fulfillment of the requirements of

Doctor of Technical Sciences

within the

Vienna PhD School of Informatics

by

Karl Stefan Gmeiner
Registration Number 0025230

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Ao.Univ.Prof. Dipl.-Inf. Dr.rer.nat. Bernhard Gramlich
Second advisor: Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Gernot Salzer

Wien, 05.02.2014

(Signature of Author)

(Signature of Advisor)

Declaration of Authorship

Karl Stefan Gmeiner
Vorgartenstraße 145–157 / 2 / 15, A - 1020 Wien, Austria

I hereby declare that I have written this Doctoral Thesis independently, that I have completely specified the utilized sources and resources and that I have definitely marked all parts of the work - including tables, maps and figures - which belong to other works or to the internet, literally or extracted, by referencing the source as borrowed.

(Place, Date) (Signature of Author)

Acknowledgements

I started programming at the age of ten, and gained a deeper understanding of algorithms at the age of sixteen when I was preparing for a programming contest. I would like to thank my classmate back then, Wolfgang Pessentheiner, for introducing me to literature on algorithms. I am sure that my studies would have turned out into a different direction without this early input.

During my studies at the TU Wien I met fellow students who gave me very valuable input and became my close friends. I would like to thank in particular Helmut Jelinek, Matthias Bauchinger, Alexander Hartmann, Dominik Schitzer, Stefan Farfeleder and all the other students with whom I collaborated in projects.

During my PhD I had a lot of interesting discussions with many colleagues at the Logic-department of the TU Wien. I would like to thank in particular Felix Schernhammer, Christoph Roschger, Eugen Jiresch, Alexander Leitsch and Gernot Salzer. I am also very grateful to Michael Reiter with whom I worked for many years.

I would like to thank the Vienna PhD-School who supported my work financially, and in particular the founder of the PhD School, Hannes Werthner. I also want to thank my colleagues during my position as a university assistant, Franz Puntigam, Andreas Krall, Anton Ertl and all others for all the interesting discussions and advises. Furthermore I want to thank everyone in our JSPS/FWF-project, in particular Naoki Nishida, for their input on this topic.

I would like to deeply thank my parents whose long (and not only financial) support allowed me to study and to pursue my studies also into different directions.

I am deeply thankful to Annu, who was my colleague at the PhD-School, then became my closest friend and later on, my wife. Without her love and gentle push during these past years, this would not have been possible.

Finally, I want to thank my supervisor Bernhard Gramlich whose lectures introduced me to term rewriting, a topic in which I found a lot of answers to my questions that arose from the very beginning of my interest in computer science. I am deeply grateful for his patience and everything I could learn from him during all these years.

Abstract

Term rewriting is a purely syntactical method to replace subterms of a term by other terms. This way, terms can be simplified until they are in normalform. Such syntactical replacements are used in many areas, for theoretical and practical purposes. Term rewriting therefore has been investigated in research for a long time and its theory is well-understood.

There are many extensions of term rewriting. One of them, conditional term rewriting comes up in many applications like for instance functional programming. Although adding conditions is an intuitive and natural approach, it makes such systems much harder to analyze. Many properties change their intuitive meaning and many criteria for properties of rewriting (like confluence or termination) do not hold anymore.

Transformations have been developed to eliminate the conditions in such conditional term rewrite systems in order to use criteria and properties of unconditional rewriting and adapt them for conditional rewriting. This approach has been used successfully for theoretical and practical purposes, yet it comes with a price.

A rewrite sequence in the transformed system should correspond to a rewrite sequence in the original system. This property is called soundness. Without soundness, the analysis of transformed systems leads to results that differ from those of the original system.

The reason why we do not obtain soundness in general is as follows: A transformation introduces some framework to encode conditions. Because of this encoding, transformed systems might give rise to different evaluation paths than the original system.

Counterexamples for soundness are usually syntactically very complex. It turns out that many of these syntactical properties are crucial for unsoundness. [14] already proves that for a certain class of conditional term rewrite systems left-linearity is a sufficient criterion to obtain soundness.

This thesis mainly deals with the question for which other syntactical properties we obtain soundness. For this purpose we present several examples that illustrate and explain in detail why we obtain unsoundness.

We then provide two different proof ideas for soundness properties. Using this approach we prove soundness for some syntactical properties, yet since these results have strong preconditions these property might be of limited use.

In the second proof approach we prove soundness for a strategy that we call U-eager rewriting. We then show how complex rewrite sequences can be transformed into such U-eager rewrite sequences. Using this approach we will prove soundness for certain linearity conditions.

Finally we will present some applications of soundness results, discuss differences of soundness properties for different transformations and provide a detailed overview of results in this thesis.

Contents

CHAPTER 1

Introduction

1.1 Term Rewriting

Term rewriting is a formal framework that has many applications in computer science. It is used for algebraic data types, in programming languages, especially functional-logic programming languages, for every kind of symbolic computation or also in theorem provers.

In term rewriting, subterms of terms are replaced by new terms according to some given set of rules, usually to simplify expressions and eventually obtain a normalform. Such repeated reductions of expressions are used to simplify mathematical expressions like $(1+2)*3$. In order to obtain a result, we use the equation $1+2=3$ and therefore replace the subterm $(1+2)$ in the original expression by 3: $(1+2)*3=3*3$. Now we apply the equation $3*3=9$ and obtain 9. This last expression 9 cannot be simplified anymore, hence it is a normalform.

In the reduction sequence, $(1+2)*3=3*3=9$, we used the equations $1+2=3$ and $3*3=9$ only from left to right to obtain a simpler expression. We would not apply it in the other direction, e.g., to "simplify" the term $(1+2)*3$ into $(1+2)*(1+2)$. We denote the direction in which we apply equations by $\rightarrow$. $(1+2)*3$ rewrites to $3*3$ using the directed equation or *rule* $1+2\rightarrow 3$, and finally $3*3\rightarrow 9$.

We can define more general rules by using placeholders or variables for which we can insert arbitrary terms. In order to simplify logic formulas we use the rule $\neg\neg A\rightarrow A$ to eliminate double negation. A here is a placeholder or *variable* for some arbitrary formula. We can simplify more complex expressions like $\neg\neg(P\vee Q)$ using this rule and obtain $P\vee Q$.

If in an expression more than one rule is applicable or more than one subterm is reducible, term rewriting does not impose any kind of determinism which subterm should be replaced or which rule should be applied. Consider the following example stemming from logic:

Example 1. *In order to convert logic expressions into conjunctive normalform we need to propagate negation inwards. By combining De Morgan's law with the rule to eliminate double*

negation we obtain the following set of rules to simplify logic formulas (A and B are variables):

$$\begin{aligned} \neg\neg A &\to A && (\neg\neg) \\ \neg(A \vee B) &\to \neg A \wedge \neg B && (DM\vee) \\ \neg(A \wedge B) &\to \neg A \vee \neg B && (DM\wedge) \end{aligned}$$

There are several rules that we can apply to the formula $\neg\neg(P \wedge \neg Q)$: By eliminating the double negation we obtain $P \wedge \neg Q$. This term is a normalform because no other rule is applicable.

We can also apply De Morgan's law and obtain $\neg(\neg P \vee \neg\neg Q)$. Again, more than one rule can be applied. All possible rewrite paths are illustrated in Figure 1.1.

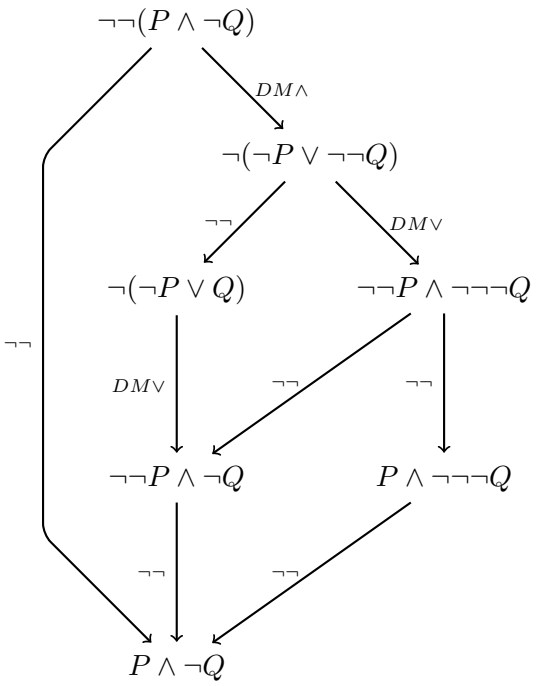

Figure 1.1: Possible rewrite paths in Example 1

Although $\neg(\neg P \vee \neg\neg Q)$ cannot be reduced to $P \wedge \neg Q$ directly, we always obtain the same normalform for this set of rules.

The previous example satisfies the following two interesting properties:

1. It is terminating (strongly normalizing), i.e., for all terms we will always reach a normalform within a finite number of steps.

2. It is confluent, i.e., if there are multiple possibilities to reduce a term, all results will eventually yield the same result.

Confluence and termination are very important and they are not always satisfied. Consider the set of rules of Example 1. If we remove the rule $DM\vee$, the start term $\neg\neg(P \wedge \neg Q)$ can be reduced to $P\wedge\neg Q$ and to $\neg(\neg P\vee\neg\neg Q)$. The latter term can be further simplified to $\neg(\neg P\vee Q)$, yet we cannot reduce this term any further, in particular we cannot simplify $\neg(\neg P \vee Q)$ to $P \wedge \neg Q$. Hence, the reduction system defined by these two rules is not confluent.

Concerning termination, consider a rule $A\vee B \to B\vee A$. Given a term $P\vee Q$, we can always apply the rule: $P \vee Q \to Q \vee P \to P \vee Q \to \cdots$ and we do not reach a normalform within a finite number of steps. Therefore, this single rule and any TRS containing it is not terminating.

We will focus on term rewriting for untyped first-order terms. Such systems have been studied for a long time and their theory is well-explored. In the past, some extensions of term rewriting have been defined for several reasons. In this thesis, we will focus on a particular extension, conditional term rewriting.

1.2 Conditional Term Rewriting

Conditional term rewriting is an extension of term rewriting in which rules are bound to certain conditions. Such conditions arise naturally, e.g. in the specification of algebraic datatypes and also in functional-logic programming. Conditional rules have a higher expressiveness than unconditional ones.

In what sense higher expressiveness?

Example 2 (Positive integers [3])**.** *We can generate all integers using the constant* 0*, the successor function* s *and the predecessor function* p*. For these functions we obtain the following rules:*

$$s(p(x)) \to x$$
$$p(s(x)) \to x$$

Our goal is to define a function pos *that returns* $true$ *for any positive integer, otherwise* $false$*. Using conditional rules we obtain the following intuitive rules:*

$$pos(0) \to false \qquad pos(s(x)) \to true \textbf{ if } pos(x) = true$$
$$pos(s(0)) \to true \qquad pos(p(x)) \to false \textbf{ if } pos(x) = false$$

We can apply a conditional rule only if a condition is satisfied. $pos(s(s(0)))$ *rewrites to* $true$ *because the condition* $pos(s(0)) = true$ *is satisfied.*

The term $pos(s(p(0)))$ *cannot be reduced using a conditional rule because the condition of the first conditional rule* $pos(p(0)) = true$ *is not satisfied.*

Although conditional rewrite rules resemble unconditional ones the condition has many consequences. In the previous example, the term $pos(p(s(0)))$ matches the left-hand side of the conditional rule $pos(s(x)) \to true$ **if** $pos(x) = true$, yet we cannot apply this rule because $pos(p(0)) \neq true$. In general, it is undecidable whether a condition is satisfiable or not (see e.g. [11]).

In the following we will use left-implication $\Leftarrow$ instead of **if**.

While conditions arise in many applications, they also are an additional source of problems. There are different possibilities how to interpret equality in the conditions. In this thesis we will usually use reducibility, meaning that s is equal to t if there is a reduction from s to t, denoted as $s \to^* t$. Furthermore, conditions may introduce variables that have not been used before. Such *extra variables* can be used to simulate `let`- or `where`-statements in functional programming.

Example 3 (Division with remainder)**.** *The functional program in Listing 1.1 divides two integers and returns the quotient and remainder.*

```
divr :: Int -> Int -> (Int, Int)

divr x y | x < y  = (0, x)
divr x y | x >= y = (q + 1, r)
                    where (q,r) = (divr (x - y) y)
```

Listing 1.1: Division with remainder in Haskell

We can encode this functional program using conditional rewrite rules. The arithmetic operations of $-$ and $<$ for natural numbers can be defined using unconditional rules:

$$\begin{aligned} x - 0 &\to x & x < 0 &\to \mathit{false} \\ 0 - y &\to 0 & 0 < s(y) &\to \mathit{true} \\ s(x) - s(y) &\to x - y & s(x) < s(y) &\to x < y \end{aligned}$$

The functional program itself is equivalent to the following conditional rewrite rules:

$$\begin{aligned} &\mathit{divr}(x, y) \to \langle 0, x\rangle \Leftarrow x < y \to^* \mathit{true} \\ &\mathit{divr}(x, y) \to \langle s(q), r\rangle \Leftarrow x < y \to^* \mathit{false}, \mathit{divr}(x - y, y) \to^* \langle q, r\rangle \end{aligned}$$

Observe that the variables q and r in the second conditional rule do not occur on the left-hand side of the rule but instead are defined in the second condition.

In order to obtain the normalform for $\mathit{divr}(s(s(s(0))), s(s(0)))$ we first need to evaluate the conditions. $s(s(s(0))) < s(s(0))$ rewrites to false, therefore we cannot apply the first conditional rule. In the second conditional rule, $\mathit{divr}(s(s(s(0))) - s(s(0)), s(s(0)))$ rewrites to $\mathit{divr}(s(0), s(s(0)))$ which rewrites to $\langle 0, s(0)\rangle$ using the first conditional rule. This corresponds to $\langle q, r\rangle$ using the variable binding 0 for q and $s(0)$ for r. Therefore, $\mathit{divr}(s(s(s(0))), s(s(0)))$ rewrites to $\langle s(0), s(0)\rangle$.

Many results of unconditional rewriting do not hold in the presence of conditions and many properties change their intuitive meaning. The Haskell program itself gives rise to an infinite recursion for the term `divr 0 0`. Nonetheless, the conditional rules in the previous example are in fact terminating because for the term $\mathit{divr}(0, 0)$ the conditions are not satisfiable.

In order to rewrite the term $\mathit{divr}(0, 0)$ using the conditional rewrite rules we first evaluate the first condition. The term $0 < 0$ evaluates to false. Hence we can apply only the second conditional rule. The second condition therefore is $\mathit{divr}(0 - 0, 0) \to^* \langle q, r\rangle$. Although it is easy to prove that this condition is unsatisfiable a rewrite engine would usually try to rewrite the term $\mathit{divr}(0, 0)$. We then obtain an infinite recursion as in the Haskell program.

For this reason we are usually only interested in *operational termination* ([13]) meaning that the whole proof tree for the conditions must be finite.

1.3 Elimination of Conditions

The analysis of conditional term rewriting is much more involved because we need to consider properties of the conditions. Also, its implementation is far from trivial because of the inherent recursion. On the other hand, unconditional term rewriting is well-understood and has many nice properties. Using transformations to map conditional rules into unconditional ones we can benefit from this fact and obtain new results for conditional rewriting and also use unconditional rewrite engines.

Several such (purely syntactical) transformations have been defined in the past for this purpose. For instance, [4] introduces a transformation to analyze confluence properties of conditional term rewriting. [5] introduces a transformation that resembles the transformations used in [14]. The latter paper analyzes a class of transformations, so-called unravelings and proves various results for this transformation.

A different class of transformations stems from [35], and is subsequently improved in [1], [32] and [6]. The latter work also presents a unified framework for properties of transformations.

In all these approaches, conditional rules are split into several unconditional rules and the conditions itself are encoded inside these unconditional rules.

Example 4. *In Example 2 we defined a function* pos *that returns true if an integer is positive and otherwise false, using the following rules:*

$$\begin{aligned} pos(0) &\to \mathit{false} & pos(s(x)) &\to \mathit{true} \Leftarrow pos(x) \to^* \mathit{true} \\ pos(s(0)) &\to \mathit{true} & pos(p(x)) &\to \mathit{false} \Leftarrow pos(x) \to^* \mathit{false} \end{aligned}$$

For the successor and predecessor function we defined the following rules:

$$\begin{aligned} s(p(x)) &\to x \\ p(s(x)) &\to x \end{aligned}$$

Consider some term that matches the left-hand side $pos(s(x))$ *of the first conditional rule. Before applying a conditional rule we need to verify the conditions. Therefore,* $pos(s(x))$ *starts a rewrite sequence of the corresponding left-hand side of the condition,* $pos(x)$*. We need to keep evaluations in the conditions separated from other rewrite steps, therefore we wrap them into a new function symbol* $\mathbf{Cond}$.[1] *Hence, we introduce a new unconditional rule*

$$pos(s(x)) \to \mathbf{Cond}(pos(x))$$

If the conditional argument rewrites to $true$*, then we reproduce the right-hand side:*

$$\mathbf{Cond}(true) \to true$$

[1] Usually in transformations $\mathbf{Cond}$ is labelled with a unique label for each rule.

In total we transform the conditional rules into the following unconditional rules:

$$pos(p(x)) \to \mathit{false} \Leftarrow pos(x) \to^* \mathit{false} \quad \begin{cases} pos(p(x)) \to \mathbf{Cond}(pos(x)) \\ \mathbf{Cond}(\mathit{false}) \to \mathit{false} \end{cases}$$

$$pos(s(x)) \to \mathit{true} \Leftarrow pos(x) \to^* \mathit{true} \quad \begin{cases} pos(s(x)) \to \mathbf{Cond}(pos(x)) \\ \mathbf{Cond}(\mathit{true}) \to \mathit{true} \end{cases}$$

We now can simulate conditional rewrite sequences using these unconditional rewrite rules:

$$pos(s(s(0))) \to \mathbf{Cond}(pos(s(0))) \to \mathbf{Cond}(\mathit{true}) \to \mathit{true}$$

Eliminating conditions is useful to improve the understanding of conditional rewriting. For instance, a conditional rewrite system with some infinite recursion immediately leads to a non-terminating unconditional rewrite system. Consider the rule $a \to b \Leftarrow f(a) \to^* b$ from [29, Section 7.2.1]. This rule is terminating because the condition is unsatisfiable but it leads to infinite recursion and therefore is not operationally terminating.

Eliminating the conditions as in Example 4 yields the following rules:

$$\begin{aligned} a &\to \mathbf{Cond}(f(a)) \\ \mathbf{Cond}(b) &\to b \end{aligned}$$

The transformed rule gives rise to the following infinite derivation:

$$a \to \mathbf{Cond}(f(a)) \to \mathbf{Cond}(f(\mathbf{Cond}(f(a)))) \to \cdots \mathbf{Cond}(f(\cdots \mathbf{Cond}(f(a)) \cdots)) \to \cdots$$

Therefore, the transformed rules are not terminating although the original rule is terminating.

1.4 Soundness of Transformations

In order to use transformations to analyze conditional term rewriting we need two properties:

1. For every conditional rewrite sequence on original terms, there is also a corresponding rewrite sequence using the transformed rules.

2. Every rewrite sequence between original terms in the transformed rewrite system corresponds to a rewrite sequence using the conditional rules.

The first property is called *completeness* (for reducibility) and is usually easy to prove. It might be surprising and counterintuitive that the second property, *soundness* (for reducibility), is not satisfied in general. In fact it turns out to be quite difficult to prove in which cases it is satisfied. We focus in this thesis on the question when a transformation is sound, in particular the transformation of [29] and [23].

One source of unsoundness are terms that represent partially evaluated conditions such that the condition is not satisfied:

Example 5 ([8, Example 32])**.** *The following conditional rule and its transformed rules represent a partial definition of the logical or:*

$$or(x,y) \to true \Leftarrow x \to^* true \quad \left\{ \begin{array}{r} or(x,y) \to \mathbf{Cond}(x) \\ \mathbf{Cond}(true) \to true \end{array} \right.$$

The terms $or(false, true)$ and $or(false, false)$ cannot be reduced using the conditional rule because the condition $x \to^* true$ is not satisfied for the variable binding $x \mapsto false$. If we add the rule $eq(x,x) \to true$ that defines a function for (syntactic) equality, the term $eq(or(false, true), or(false, false))$ cannot be simplified, in particular we cannot rewrite it to $true$.

In the transformed system, we need to verify (or disprove) the conditions by introducing the conditional arguments:

$$or(false, true) \to \mathbf{Cond}(false)$$
$$or(false, false) \to \mathbf{Cond}(false)$$

The term $\mathbf{Cond}(false)$ cannot be reduced any further because the condition is not satisfiable. Yet, although both or-terms are different and do not have a common successor using the original rules, they have a common successor using the transformed rules.

Using the rule $eq(x,x) \to true$, the transformed unconditional rules give rise to the following derivation:

$$\begin{aligned} eq(or(false, true), or(false, false)) &\to eq(\mathbf{Cond}(false), or(false, false)) \\ &\to eq(\mathbf{Cond}(false), \mathbf{Cond}(false)) \to true \end{aligned}$$

Using the conditional rules we cannot repeat this equality proof. Therefore the transformation is not sound.

Even without the non-left-linear rule $eq(x,x) \to true$ we obtain another unsoundness property, unsoundness w.r.t. joinability, because both or-terms have a common successor if we apply the transformed rules. Nevertheless using the conditional rules, both of them are already irreducible.

We introduced the function symbol $\mathbf{Cond}$ in order to wrap the condition and thereby distinguish the evaluation of a condition from non-conditional rewrite steps. Mostly, we do not know in advance whether a condition is satisfiable or not. In the previous example, the conditions are not satisfied and we obtain some garbage $\mathbf{Cond}$-terms that give rise to unsoundness in connection with a non-left-linear rule.

We cannot avoid failed rewrite attempts in the conditional arguments. Yet, we can avoid that terms that represent failed attempts contain less information than the term that initiated the conditional evaluation. For this purpose, we replace the $\mathbf{Cond}$-symbol by a new symbol U (after the name of this class of transformations, *unravelings* [14, Definition 3.1]). We extend this symbol by a unique labeling for every conditional rule and add all variable bindings of the left-hand side of the conditional rule as arguments. We then obtain a transformation that resembles the unraveling of [14, Definition 6.1].

The conditional rule of Example 5 then is transformed into the following rules (α is a unique label for this conditional rule):

$$or(x,y) \to true \Leftarrow x \to^* true \quad \left\{ \begin{array}{r} or(x,y) \to U_\alpha(x,x,y) \\ U_\alpha(true,x,y) \to true \end{array} \right.$$

The first argument of U_α contains the condition while the other arguments store the variable bindings.

Using the transformed rules we obtain the following derivations:

$$or(false,true) \to U_\alpha(false,false,true)$$
$$or(false,false) \to U_\alpha(false,false,false)$$

Both U-terms are irreducible, but they are not syntactically equal. Hence we cannot reproduce unsoundness as in Example 5.

If the U-terms contain all variable bindings of the left-hand side of a conditional rule, we can use them to map a U-term to an original term: We extract these variable bindings from the U-term and insert it into the left-hand side of the conditional rule. For instance, the U-symbol U_α is introduced by transforming the conditional rule $or(x,y) \to true \Leftarrow x \to^* true$. Therefore, by extracting the variable bindings of the U-term $U_\alpha(false,false,true)$ and inserting them into the left-hand side $or(x,y)$ of the conditional rule we obtain $or(false,true)$. We will refer to this mapping as translation-backwards or backtranslation.

Such mappings will play an important role in our soundness proofs because we can translate U-terms into original terms. Yet, translating rewrite sequences in a transformed system using this approach might not yield valid derivations of the original system in general because of incompatible rewrite steps in the condition and the variable binding:

Example 6. *Consider the unconditional rules*

$$a \to c$$
$$a \to d$$
$$s(c) \to t(k)$$

and the following conditional rule and its transformation using our previous unraveling:

$$f(x) \to z \Leftarrow s(x) \to^* t(z) \quad \left\{ \begin{array}{r} f(x) \to U_\alpha(s(x),x) \\ U_\alpha(t(z),x) \to z \end{array} \right.$$

The U-symbol has two arguments: The first one contains the conditional evaluation, the second one the variable binding.

We obtain the following derivation using the transformed rules:

$$f(a) \to U_\alpha(s(a),a) \to U_\alpha(s(c),a) \to U_\alpha(t(k),a) \to U_\alpha(t(k),d) \to k$$

The second argument contains the variable binding used in the left-hand side of the conditional rule. Replacing U-terms by the corresponding left-hand side of the conditional rule by inserting the variable binding yields the following sequence:

$$f(a) \;\to\; \underbrace{U_\alpha(s(a),a)}_{f(a)} \;\to\; \underbrace{U_\alpha(s(c),a)}_{f(a)} \;\to\; \underbrace{U_\alpha(t(k),a)}_{f(a)} \;\to\; \underbrace{U_\alpha(t(k),d)}_{f(d)} \;\to\; k$$

The last rewrite step of the derivation would correspond to the rewrite step $f(d) \to k$ using the conditional rules. Yet, the condition $s(d) \to^* t(k)$ is not satisfied. Therefore this rewrite step is not possible in the original system.

Observe, that the whole derivation is not unsound because of $f(a) \to^* k$ using the conditional rules.

We can exploit such incompatibilities of the conditional argument and the variable bindings with a sophisticated combination of non-linear rewrite rules. [14] provides such a counterexample for soundness (Example 20).

In [31] it was shown that we obtain soundness (and completeness) by preventing rewrite steps in the variable bindings. For unrestricted rewriting, we can prove that certain syntactical properties prevent such inconsistencies that cause unsoundness. In [14] and [29] it was shown that left-linearity of all transformed rules is sufficient for soundness for a certain unraveling. We showed and disproved soundness for several other properties in [7], in particular we proved that for certain conditional rewrite systems confluence, non-erasingness and weak left-linearity are sufficient for soundness. Especially the latter result was a major improvement of other soundness results. We managed to adapt some of our results in [8] for another unraveling ([29], based on [15]) for so-called deterministic conditional term rewrite systems, a class of conditional rewrite systems that allow extra variables to a certain extend.

Based on this unraveling, [24] introduces an unraveling that optimizes the use of variable arguments. It is shown that for this optimized unraveling a combination of context-sensitive rewriting and membership conditions is sufficient for soundness. In [25] it is shown that also left-linearity, and right-linearity and non-erasingness are sufficient for soundness. Another result is that soundness of this optimized unraveling implies soundness of the unraveling in [29].

Further soundness properties and similar properties for other transformations are shown in e.g. [32].

From a practical point of view, soundness plays an important role whenever conditional rewrite sequences are simulated using unconditional rules:

- In [24] unravelings are used to (partially) invert functions that are defined as a term rewrite system. For this application, soundness is highly desirable since otherwise the inverted function may return wrong results.

- Related to the last paper, [22] provides results on proving injectivity of functions by completion of unraveled conditional rewrite systems.

- In [32] a transformation is introduced that yields "computational equivalence, i.e., that can be used to simulate conditional rewrite sequences without the need for backtracking.

- In [9] we prove confluence of conditional rewrite systems using unravelings. In order to do this we need to prove soundness first.

- Soundness is essential to prove non-(operational)-termination using transformations. [31] investigates this point in more detail.

1.5 Contributions

The main contribution of our work is the detailed analysis of soundness properties of the unraveling of [29].

- We analyze counterexamples for soundness in detail. We show that the variables in the conditions of conditional rules must be distributed in a certain way (Example 18). This leads to the definition of a variable property, sortedness. Although it is not sufficient for soundness it is an important precondition to avoid unsoundness in many cases. We then analyze the unsoundness example of [14] and present a new unsoundness example in which we show various unsoundness properties and how they depend on each other (Example 21).

- Based on these counterexamples we show in which cases we can prove soundness using just syntactic term translations. We prove soundness for right-separated 2-DCTRSs if the transformed system is non-erasing for the unraveling of [24] (Theorem 1). This result is in particular interesting because it is not satisfied for other transformations (see e.g. Example 43). We then also prove soundness for left-separated CTRSs (Theorem 2). This result corresponds to the case of ultra-right-linear CTRSs in [7] and [8]. Using the same proof idea we finally prove soundness w.r.t. joinability for confluent, sorted DCTRSs.

- We introduce the class of U-eager derivations in which terms that encode conditions must be immediately rewritten until the condition is satisfied. We then prove soundness of this class for the transformation of [24] (Lemma 14). By converting other derivations into such U-eager derivations we obtain further soundness properties.

- We show that we obtain soundness also for the transformation of [29] using our previous approach. We prove a not yet published soundness result for right-linear DCTRSs (Theorem 6). Finally we show a new proof for our main result of [8], soundness of weakly left-linear DCTRSs 8.

Finally we will discuss applications of transformations. Parts of this work are based on [9] where we showed how to prove confluence of CTRSs using unravelings. For this purpose, another soundness property, soundness w.r.t. joinability (Theorem 3, Theorem 7, Theorem 9) is of importance.

Finally, we provide examples to illustrate differences to soundness properties of other transformations, discuss related work and summarize our results.

1.6 Outline

In Chapter 2 we will provide an overview on notions and notations in abstract and (conditional) term rewriting where we mostly follow [3] and [29].

In Chapter 3 we introduce the simultaneous unraveling of [14], the sequential unraveling of [29] and the optimized unraveling of [24]. We introduce the notion of ultra-properties and term translations for these unravelings.

Our main chapter, Chapter 4, is divided into several sections. In the beginning we will provide examples for unsoundness and define the notion of sortedness that restricts variable occurrences in the conditions. We show that although sortedness is not sufficient for soundness it avoids many trivial examples.

After presenting a new unsoundness example we prove soundness properties by translating derivations in transformed systems directly. This approach allows us to prove soundness for many examples, yet it requires severe syntactical restrictions.

Then we show soundness for a certain rewrite strategy, U-eager rewrite sequences. This result will be the foundation for all further soundness results. By shifting rewrite steps in derivations we will show how to transform other rewrite sequences into such U-eager derivations.

We will show the usefulness of this approach at the end of Chapter 4. There we will introduce a mapping from the sequential unraveling of [29] to the optimized unraveling. We will prove that for certain linearity properties we can apply this mapping and thereby prove soundness for the sequential unraveling. Finally we present confluence results for conditional term rewrite systems based on our results.

In Chapter 5 we will provide further applications of soundness results. We also discuss related work and provide counterexamples for soundness of other transformations. Finally we provide an overview of the most important results in this thesis.

CHAPTER 2

Preliminaries

2.1 Abstract Reduction Systems (ARSs)

An *abstract reduction system* (*ARS*) is a pair $\mathcal{A} = (A, \to)$ of some set A and a binary relation $\to$ over A. We usually write $a \to b$ instead of $(a, b) \in \to$, which means that a is reduced to b. $\to$ is the *reduction* or *rewrite relation*.

By composing multiple (possibly infinitely many) reduction steps we obtain a *reduction sequence* or *derivation* $a_1 \to a_2 \to \cdots$. If a reduces to b using a reduction sequence of length $n + 1$, then $a \to^{n+1} b$. The relation $\to^{n+1}$ is defined as $\to^n \circ \to$ where $\to^0 = \{(a, a) \mid a \in A\}$ is the identity on A.

$\to^{\leq n} = \bigcup_{i=0}^{n} \to^i$ is the union of all compositions of length n or less. The reflexive closure of $\to$ is denoted as $\to^{\leq 1}$ or $\to^{=}$. The transitive closure is $\to^+ = \bigcup_{i>0} \to^i$, the reflexive transitive closure is $\to^* = \to^0 \cup \to^+$.

The inverse of $\to$ is $\leftarrow = \{(b, a) \mid (a, b) \in \to\}$.

The symmetric closure of $\to$ is $\leftrightarrow = \leftarrow \cup \to$.

The reflexive, transitive and symmetric closure is $\leftrightarrow^* = (\leftrightarrow)^*$.

If $a \to^+ b$ ($a \to b$), then b is a (direct) *successor* of a, and a is a (direct) *predecessor* of b.

Some $a \in A$ is *reducible* if there is a $b \in A$ such that $a \to b$. It is *irreducible* or a *normalform* if there is no such b. b is a normalform of a if $a \to^* b$ and b is a normalform.

a and b are *joinable* ($a \downarrow b$) if there is a $c \in A$ such that $a \to^* c$ and $b \to^* c$.

An ARS is *strongly normalizing* or *terminating* if for all $a_0 \in A$ there is no infinite reduction sequence $a_0 \to a_1 \to \cdots$.

It is *confluent* if $a \to^* b$ and $a \to^* c$ implies $b \downarrow c$. It is *weakly confluent* if $c \leftarrow a \to b$ implies $b \downarrow c$. Newman's Lemma states that weak confluence and termination imply confluence.

We call an ARS *convergent* if it is confluent and terminating.

We will refer to relation $\to^*$ as *reducibility*, to $\downarrow$ as *joinability* and to $\leftrightarrow^*$ as *convertibility*.

2.2 Terms and Substitutions

2.2.1 Terms

A signature $\mathcal{F}$ is a set of function symbols that are associated with a non-negative integer. This integer represents the *arity* of a function symbol f, denoted as $\mathsf{ar}(f)$. The set of all function symbols with arity n in the signature $\mathcal{F}$ is $\mathcal{F}^{(n)}$.

A function symbol with arity 0 is a *constant*. We will usually denote constants as $a, b, c, \ldots$ and other functions as $f, g, h, \ldots$. In the following we will keep the arity of function symbols implicit.

$\mathcal{V}$ is a countably infinite set of variables. We assume that $\mathcal{F} \cap \mathcal{V} = \emptyset$. We denote variables usually as $x, y, z, \ldots$.

A term is a tree structure in which nodes are labelled by function symbols and variables. In this thesis, variables only can appear as leaf nodes. The number of children of a node is equivalent to the arity of its labelling function symbol. $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is the set of terms over the signature $\mathcal{F}$ and inductively defined as follows:

$$\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$$
$$f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \quad \text{where } f \in \mathcal{F}^{(n)} \text{ and } t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$$

The mapping $\mathsf{root} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \mapsto \mathcal{F} \cup \mathcal{V}$ returns the root symbol of a term

$$\mathsf{root}(s) = \begin{cases} s & \text{if } s \in \mathcal{V} \\ f & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

We will call terms with root symbol f f-terms. The child nodes of a term are its arguments.

The set of all variables in a term $\mathcal{V}ar(s)$ is

$$\mathcal{V}ar(s) = \begin{cases} \{s\} & \text{if } s \in \mathcal{V} \\ \bigcup_{i=1}^{n} \mathcal{V}ar(s_i) & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

A term s is *ground* if $\mathcal{V}ar(s) = \emptyset$.

The number of occurrences of a variable $x \in \mathcal{V}ar(s)$ is

$$|s|_x = \begin{cases} 1 & \text{if } s = x \\ 0 & \text{if } s \neq x \text{ and } s \in \mathcal{F}^0 \cup \mathcal{V} \\ \Sigma_{i=1}^{n} |s_i|_x & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

A term s is *linear* if for all $x \in \mathcal{V}ar(s)$, $|s|_x = 1$. If $|s|_x > 1$, then x is a non-linear variable in s, otherwise it is a linear variable.

2.2.2 Term Positions

The set of all term positions of a term s is a set of strings over the alphabet of positive integers:

$$\mathcal{P}os(s) = \begin{cases} \{\epsilon\} & \text{if } s \in \mathcal{V} \cup \mathcal{F}^0 \\ \{\epsilon\} \cup \{i.p_i \mid 1 \leq i \leq n, p_i \in \mathcal{P}os(s_i)\} & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

The subterm of the term s at position $p \in \mathcal{P}os(s)$ denoted as $s|_p$ is defined recursively:

$$s|_p = \begin{cases} s & \text{if } p = \epsilon \\ s_i|_{p'} & \text{if } p = i.p' \text{ and } s = f(s_1, \ldots, s_n) \end{cases}$$

The set of all *variable positions* in s is $\mathcal{VP}os(s) = \{p \in \mathcal{P}os(s) \mid s|_p \in \mathcal{V}\}$. The set of non-variable positions of s is $\mathcal{FP}os(s) = \mathcal{P}os(s) \setminus \mathcal{VP}os(s)$. We sometimes refer to a non-variable position p in s as an f-position if $f = \mathsf{root}(s|_p)$.

$|p|$ denotes the length of the string. The empty string ϵ represents the root position.

The position p is above the position q ($p, q \in \mathcal{P}os(s)$) if there is a p' such that $p.p' = q$ (where . represents the concatenation of strings). It is strictly above q if $p' \neq \epsilon$. If p is (strictly) above q, then p is a (proper) prefix of q, denoted as $p \leq q$ ($p < q$). If $p \geq q$ ($p > q$), p is (strictly) below q.

p and q are parallel positions, $p \parallel q$, if neither p is above q nor p is below q.

$s[t]_p$ is the term in which the subterm at position p in s has been replaced by t:

$$s[t]_p = \begin{cases} t & \text{if } p = \epsilon \\ f(s_1, \ldots, s_{i-1}, s_i[t]_{p'}, s_{i+1}, \ldots, s_n) & \text{if } p = i.p' \text{ and } s = f(s_1, \ldots, s_n) \end{cases}$$

This notion is extended to sets of positions: $s[t]_{\{p_1,\ldots,p_k\}} = s[t]_{p_1} \ldots [t]_{p_k}$ ($p_i \parallel p_j$ for all $1 \leq i, j \leq k$ and $i \neq j$). We furthermore shorten multiple consecutive replacements $s[t_1]_{p_1} \ldots [t_n]_{p_n}$ to $s[t_1, \ldots, t_k]_{p_1,\ldots,p_k}$.

A context C is a term over the signature $\mathcal{F}$ plus a special constant $\Box \notin \mathcal{F} \cup \mathcal{V}$ called *hole*, $C \in \mathcal{T}(\mathcal{F} \cup \{\Box\}, \mathcal{V})$. $C[t_1, \ldots, t_k]$ denotes the term in which all holes in C from left to right (relative to the tree representation of C) have been replaced by $t_1, \ldots, t_k$.

For a finite set of terms $M \subset \mathcal{T}$, $\overrightarrow{M}$ is a sequence of all terms in M in an unspecified but fixed order. $\overrightarrow{M}[i]$ is the ith term in this sequence.

2.2.3 Substitutions

A substitution σ is a function $\mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ such that $\sigma(x) \neq x$ for finitely many $x \in \mathcal{V}$. The domain of a substitution σ is the set of only these variables $\mathcal{D}om(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$.

Given a substitution σ and a set of variables $X \subset \mathcal{V}$, $\sigma\!/_X$ denotes the substitution

$$\sigma\!/_X(x) = \begin{cases} \sigma(x) & \text{if } x \in X \\ x & \text{otherwise} \end{cases}$$

In the following we will use the common postfix notation $x\sigma$ instead of $\sigma(x)$.

The range of a substitution is the set of all images of σ, $\mathcal{R}an(\sigma) = \{x\sigma \mid x \in \mathcal{D}om(\sigma)\}$. The variable range is the set of all variables in the range $\mathcal{VR}an(\sigma) = \bigcup_{x \in \mathcal{D}om(\sigma)} \mathcal{V}ar(x\sigma)$. In the following we usually assume w.l.o.g. that $\mathcal{D}om(\sigma) \cap \mathcal{VR}an(\sigma) = \emptyset$.

The composition of substitutions $\sigma \circ \tau$ is defined as $x(\sigma \circ \tau) = (x\sigma)\tau$.

A substitution σ is more general than a substitution τ if there is a substitution σ' such that $\sigma \circ \sigma' = \tau$.

Substitutions are extended homomorphically to terms.

$$s\sigma = \begin{cases} x\sigma & \text{if } s = x \text{ and } x \in \mathcal{V} \\ f(s_1\sigma, \ldots, s_n\sigma) & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

A term s matches a term t if there is a substitution σ such that $s\sigma = t$. In this case, t is an *instance* of s and σ is a *matcher*.

Two terms s, t are unifiable if there is a substitution σ such that $s\sigma = t\sigma$; σ is called a *unifier* of s and t. σ is a *most general unifier* if it is more general than all other unifiers of s and t.

2.3 Term Rewrite Systems (TRSs)

A *term rewrite rule* (or simply rewrite rule) is a pair of terms l, r, denoted as $l \to r$, such that l is not a variable and $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$.

A *term rewrite system* (*TRS*) is a pair of a signature $\mathcal{F}$ and a set of rewrite rules R, $\mathcal{R} = (\mathcal{F}, R)$. By abuse of notation we will use $\mathcal{R}$ and R interchangeably and usually leave the signature implicit.

A rule $l \to r$ is *left-linear* if l is linear, and *right-linear* if r is linear. It is *non-erasing* if $\mathcal{V}ar(l) \setminus \mathcal{V}ar(r) = \emptyset$, and *erasing* otherwise. In the latter case, all variables in $\mathcal{V}ar(l) \setminus \mathcal{V}ar(r)$ are *erased variables* and terms that are bound to these variables are *erased*. A rule is *collapsing* if its right-hand side is a variable ($r \in \mathcal{V}$), otherwise it is *non-collapsing*.

A TRS is left-linear (right-linear, non-erasing, non-collapsing) if all its rules are left-linear (right-linear, non-erasing, non-collapsing). It is erasing (collapsing) if one of its rules is erasing (collapsing).

The set of *defined symbols* $\mathcal{D} = \{\mathsf{root}(l) \mid l \to r \in \mathcal{R}\}$ is the set of all function symbols in $\mathcal{F}$ that are at the root position of the left-hand side of some rule. All other function symbols are *constructor symbols* $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$. $\mathcal{T}(\mathcal{C}, \mathcal{V})$ is the set of constructor terms. A TRS is a *constructor system* if the left-hand side of every rule has the form $f(s_1, \ldots, s_n)$ where $s_1, \ldots, s_n \in \mathcal{T}(\mathcal{C}, \mathcal{V})$.

Let $\mathcal{R} = (\mathcal{F}, R)$ be a term rewrite system. An instance of some left-hand side of a rule is a *reducible expression* (*redex*). The rewrite relation $\to_{\mathcal{R}}$ of a TRS $\mathcal{R}$ is defined as follows: If $(s, t) \in \to_{\mathcal{R}}$, then there is a rule $l \to r$ and a position $p \in \mathcal{P}os(s)$ such that $s|_p = l\sigma$ is a redex. The redex is replaced by (contracted into) the corresponding instance of the right-hand side in t, and in $t = s[r\sigma]_p$. t is a *reduct* of s.

We usually denote a rewrite step as $s \to t$ where we annotate $\to$ with some additional information like the position of the redex and the applied rule if necessary.

In the parallel rewrite relation $\Vdash$ we may apply multiple rewrite steps at once provided that all redexes occur in parallel positions:

$$s \Vdash_\emptyset s$$
$$s \Vdash_{\{q\} \cup P} t \quad \text{if } s \to_q u, u \Vdash_P t \text{ and } q \parallel p \text{ for all } p \in P$$

We trace term positions in derivations using *descendants* and *ancestors*. In a rewrite step $s \to_{p,l\to r} t$, the set of *one-step-descendants* of some $q \in \mathcal{P}os(s)$ is

$$\begin{array}{ll} \{q\} & \text{if } q \parallel p \text{ or } q < p \\ \{p.p_3.q_2 \mid r|_{p_3} = l|_{p_1}\} & \text{if } q = p.p_1.p_2 \text{ where } p_1 \in \mathcal{VP}os(l) \\ \emptyset & \text{otherwise} \end{array}$$

If o is a one-step-descendant of q, then q is a *one-step-ancestor* of o. Observe that the contracted redex is not a one-step-descendant of the redex.

By slight abuse of notation, we will sometimes refer to $t|_q$ as a one-step-descendant of the term $s|_p$ if q is a one-step descendant of p.

The descendant relation (ancestor relation) is defined as the reflexive, transitive closure of the one-step-descendant (one-step-ancestor) relation.

Let $\alpha : l_1 \to r_1$ and $\beta : l_2 \to r_2$ be two rules in a TRS where all variables in β have been renamed such that $\mathcal{V}ar(\alpha) \cap \mathcal{V}ar(\beta) = \emptyset$. α and β are *overlapping* if there is a $p \in \mathcal{FP}os(l_1)$ such that $l_1|_p$ and l_2 are unifiable, i.e., $l_1|_p\sigma = l_2\sigma$. An overlap is a self-overlap if α and β are instances of the same rule.

The pair $(r_1\sigma, l_1[r_2\sigma]_p)$ is a *critical pair* if $l_1|_p$ and l_2 give rise to an overlap with the most general unifier σ that is not a self-overlap at root position.

A TRS is weakly confluent if all its critical pairs are joinable.

A TRS is overlapping if its rules give rise to critical pairs. It is an *overlay system* if all overlaps are at root position.

A TRS is *orthogonal* if it is non-overlapping and left-linear. Every orthogonal TRS is confluent.

2.3.1 Context-sensitive term rewriting

A mapping $\mu : \mathcal{F} \mapsto \mathcal{P}(\mathbb{N})$ is a *replacement map* if for all $f \in \mathcal{F}$, $\mu(f) \subseteq \{1, \ldots, \mathsf{ar}(f)\}$. The set of μ-replacing positions is defined as follows:

$$\mathcal{P}os^\mu(s) = \begin{cases} \{\epsilon\} & \text{if } s \text{ is a variable} \\ \{\epsilon\} \cup \bigcup_{i\in\mu(\mathsf{root}(s))} \{i.p \mid p \in \mathcal{P}os^\mu(s|_i)\} & \text{otherwise} \end{cases}$$

In *context-sensitive term rewriting*[12] a rewrite step $s \to_{p,\mathcal{R}} t$ is also a μ-rewrite step $s \hookrightarrow_\mu t$ if $s|_p$ is a μ-replacing redex, i.e., $p \in \mathcal{P}os^\mu(s)$.

2.3.2 Membership-conditional term rewriting

In membership-conditional term rewriting [34] rules are bound to membership conditions:

$$\alpha : l \to r \Leftarrow x_1 \in \mathcal{T}_1, \ldots, x_k \in \mathcal{T}_k$$

We can apply this rule to a redex $l\sigma$ if the variable bindings satisfy the membership condition, $x_i\sigma \in \mathcal{T}_i$ for all $i \in \{1, \ldots, k\}$.

We will use a simplified membership condition that is also used in [19]: Given a TRS $\mathcal{R}$ and a set of terms $\mathcal{T}' \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$, a rewrite step $s \to_{p,l\to r,\mathcal{R}} t$ satisfies the membership condition $\in \mathcal{T}'$, denoted as $s \hookrightarrow_{\in\mathcal{T}'} t$, if $\mathcal{R}an(\sigma) \subseteq \mathcal{T}'$ where σ is the matcher of the redex $s|_p = l\sigma$.

2.3.3 Conditional Term Rewrite Systems (CTRSs)

A conditional term rewrite rule is a triple (l, r, c) where $l, r \in \mathcal{T}(\mathcal{F}, \mathcal{V})$. We will denote conditional rules as $l \to r \Leftarrow c$. The condition c is usually a (possibly empty) conjunction of equations $s_1 = t_1, \ldots, s_k = t_k$.

A conditional term rewrite system (CTRS) consists of conditional rules. The *underlying TRS* of a CTRS $\mathcal{R}$ is Clarify "TRS" because this one might not be one in 3-CTRSs. $\mathcal{R}_u = \{l \to r \mid l \to r \Leftarrow c \in \mathcal{R}\}$. A CTRS is left-linear, right-linear, (non-)erasing, (non-)overlapping, an overlay system, (non-)collapsing, a constructor system or orthogonal if $\mathcal{R}_u$ is.

In contrast to unconditional term rewrite systems, conditional rewrite rules may introduce new variables in the conditions and on their right-hand side. Variables in a conditional rule that do not occur in the left-hand side are *extra-variables*, $\mathcal{EV}ar(l \to r \Leftarrow c) = \mathcal{V}ar(r, c) \setminus \mathcal{V}ar(l)$.

We can classify conditional rewrite rules according to the distribution of extra variables as in [16]. Table 2.1 shows all types of conditional rules.

Table 2.1: Types of the conditional rewrite rule $l \to r \Leftarrow c$ [16]

Type 1	If $\mathcal{V}ar(r) \cup \mathcal{V}ar(c) \subseteq \mathcal{V}ar(l)$	(no extra variables)
Type 2	If $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$	(extra variables may only occur in conditions)
Type 3	If $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l) \cup \mathcal{V}ar(c)$	(all extra variables occur in the conditions)
Type 4	No restrictions	

A CTRS is an n-CTRS if all its rules are of type n.

Depending on the interpretation of equality we obtain different rewrite relations. Table 2.2 contains possible interpretations ([4]). We use the notions that are used in [29]. In this thesis we will mostly focus on oriented CTRSs.

Table 2.2: Classes of conditional rewrite rules [29]

Semi-equational CTRS	$s = t$ is equivalent to $s \leftrightarrow^* t$
Join CTRS	$s = t$ is equivalent to $s \downarrow t$
Oriented CTRS	$s = t$ is equivalent to $s \to^* t$
Normal CTRS	$s = t$ is equivalent to $s \to^* t$ and t is an irreducible ground term w.r.t. $\mathcal{R}_u$

A CTRS is *right-stable* if $\mathcal{V}ar(l, s_1, \ldots, s_i, t_1, \ldots, t_{i-1}) \cap \mathcal{V}ar(t_i) = \emptyset$ and all $t_1, \ldots, t_k$ are linear constructor terms for all rules $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$.

An extra variable z in a rule $\alpha : l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ is a *deterministic extra variable*, if it first occurs on the right-hand side of a condition: $z \in \mathcal{V}ar(t_i) \setminus \mathcal{V}ar(s_1, t_1, \ldots, s_{i-1}, t_{i-1}, s_i)$ for some $i \in \{1, \ldots, k\}$.

A conditional rewrite rule is *deterministic* if all its extra variables are deterministic, i.e.,

$$\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_{i-1}) \text{ and } \mathcal{V}ar(r) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_k)$$

A *deterministic CTRS* (DCTRS) is an oriented 3-CTRS such that every rule is deterministic.

The rewrite relation of an oriented CTRS $\mathcal{R}$ is defined inductively. We define the unconditional TRSs $\mathcal{R}_n$: Again clarify TRS here.

$$\mathcal{R}_0 = \emptyset$$
$$\mathcal{R}_{n+1} = \left\{ \begin{array}{r} l\sigma \to r\sigma \mid l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k \in \mathcal{R} \\ \text{and } s_i\sigma \to^*_{\mathcal{R}_n} t_i\sigma \text{ for all } i \in \{1, \ldots, k\} \end{array} \right\}$$

The rewrite relation $\to_{\mathcal{R}}$ of the CTRS $\mathcal{R}$ is $\to_{\mathcal{R}} = \bigcup_{n \in \mathbb{N}} \to_{\mathcal{R}_n}$.

The *depth* of a conditional rewrite step $s \to_{\mathcal{R}} t$ is the smallest n such that $s \to_{\mathcal{R}_n} t$. $depth(s \to^*_{\mathcal{R}} t)$ is the maximum depth of all rewrite steps in the rewrite sequence $s \to^* t$.

CHAPTER 3

Unravelings

3.1 Transformations of CTRSs into TRSs

3.1.1 Introduction

Conditional term rewrite systems (CTRSs) are a natural extension of unconditional ones (TRSs). Yet, the inherent complexity that is caused by the recursive evaluation of the conditions makes it difficult to analyze them because most non-syntactic properties depend on whether a conditional rule is applicable or not. Several properties that are satisfied for unconditional TRSs are therefore not satisfied for CTRSs.

Example 7 ([4, Example 3.6])**.** *The following join 1-CTRS that is based on [4, Example 3.6] is non-overlapping and left-linear and therefore orthogonal.*

$$\mathcal{R} = \left\{ \begin{array}{c} f(x) \to b \Leftarrow f(x) \downarrow x \\ a \to f(a) \end{array} \right\}$$

Yet, it is not confluent: $f(a)$ *rewrites to* b *because the condition* $f(a) \downarrow a$ *is satisfiable. Furthermore,* $f(a) \to f(f(a)) \to f(b)$*. Yet,* $f(b)$ *is irreducible because the condition* $f(b) \downarrow b$ *is not satisfiable. Therefore,* b *and* $f(b)$ *are not joinable. Hence,* $\mathcal{R}$ *is not confluent.*

Other properties of unconditional term rewriting depend on the decidability whether a rule is applicable or not. These properties do not properly take into account the additional complexity of evaluating the conditions. Termination only implies the absence of infinite rewrite paths until we reach a normalform. In unconditional rewriting this implies that we will reach a normalform by continuous rewriting using finitely many rewrite steps. In conditional rewriting a rewrite sequence might consist of finitely many conditional rewrite steps, yet one single conditional rewrite step might cause infinitely many rewrite steps in the conditions or infinite recursion steps. Termination therefore does not imply that we reach a normalform in finite time for conditional rewriting.

Example 8 ([13]). *Consider the single conditional rule $a \to b \Leftarrow f(a) \to^* b$. The condition is obviously not satisfied, therefore the rewrite relation of this single rule is empty. The CTRS is therefore terminating. It is even effectively terminating as is pointed out in [29] which means that the rewrite relation is effectively computable.*

Practically, in order to show that a is irreducible we would attempt to rewrite a using the only rewrite rule. Therefore, we need to evaluate the condition $f(a) \to^* b$. Yet, when we try to evaluate $f(a)$ we immediately end up trying to rewrite a and therefore we obtain a loop.

For CTRSs termination loses its intuitive meaning of the absence of infinite evaluation parts. In [13] it is therefore proposed to introduce a new termination notion, so-called *operational termination* that implies that all proof attempts must terminate successfully or fail within a finite number of steps. This notion therefore also includes recursive steps and rewrite steps in the conditions.

From a practical point of view, implementing conditional term rewriting is far from being trivial, not only because of the recursive evaluations of conditions but also because we should somehow cache failed conditional evaluations for efficiency reasons.

Transforming systems into equivalent, simpler ones is a common approach in many fields for both theoretical and practical reasons. It is therefore intuitive to try to develop a transformation from conditional rewrite systems into unconditional ones.

In order to benefit from such a transformation we need to prove properties of these transformations themselves. [4] contains one such transformation that slightly resembles the unraveling for normal CTRSs that is introduced in [14], yet it is only used to construct counterexamples for confluence properties.

3.1.2 Overview

In the following we will mainly consider oriented CTRSs. This is due to the fact that we can simulate a join CTRS by an oriented CTRS by translating join conditions $s \downarrow t$ into two oriented conditions $s \to^* z, t \to^* z$ where z is a new extra variable.

Unravelings

Although transformations of CTRSs have been used before (e.g. [4,5,10]), [14] is the first work known to us that mainly focuses on properties of transformations of conditional term rewrite systems. In this work, so-called *unravelings* are introduced, a class of transformations.

In [14, Definition 3.1], a transformation is an unraveling $\mathbb{U}$ if it is complete for joinability ($\downarrow_{\mathcal{R}} \subseteq \downarrow_{\mathbb{U}(\mathcal{R})}$) and modular w.r.t. unconditional rules ($\mathbb{U}(\mathcal{R} \cup T) = \mathbb{U}(\mathcal{R}) \cup T$ where T is a set of unconditional rules).

We will refer to all transformations as unravelings that preserve the original signature ($\mathcal{F} \subseteq \mathbb{U}(\mathcal{F})$ where $\mathcal{F}$ is the original signature while $\mathbb{U}(\mathcal{F})$ is the signature of the transformed system).

In [14], two unravelings are presented, one for join 1-CTRSs and one for normal 1-CTRSs. In both unravelings, one new function symbol is introduced for each conditional rule. In this function symbol, all conditions of the rule are encoded simultaneously, along with some vari-

ables to avoid extra variables in the transformed system. In the following we will call this transformation *simultaneous unraveling*.

In [15] this approach is extended to deterministic CTRSs, yet without formal proofs. The definition of the transformation considers the iterative nature of deterministic extra variables and therefore encodes the conditions not simultaneously but sequentially instead. For this purpose, one new function symbol is introduced for every condition. Again, some variable bindings are also encoded to avoid extra variables in the transformed system. The unraveling is slightly refined in [27]. In the following we will call this refined transformation the *sequential unraveling*.

In [7] we have proven several new interesting results for the simultaneous unraveling.

In [25], the sequential unraveling is optimized by minimizing the number of variable bindings. Here, we will refer to this transformation as *optimized unraveling*.

In [8] we extend our analysis results of [7] to the sequential unraveling and the optimized sequential unraveling or disproved them.

Other Transformations

In [1] a transformation is introduced that is based on [35]. In this transformation, root symbols of the left-hand side of conditional rules are replaced by new function symbols with a higher arity. These additional arguments are then used to encode the conditions. Since multiple conditions are encoded in parallel positions we can evaluate them in parallel.

In this transformation we need to replace unconditional rules by equivalent rules using the new signature so that the transformation does not satisfy the condition $\mathbb{U}(\mathcal{R} \cup T) = \mathbb{U}(\mathcal{R}) \cup T$. Hence this transformation and transformations that are based on this approach are not unravelings.

Example 9 ([1, Example 5])**.** *The following conditional rules define the function* abs *that returns the absolute value of a number:*

$$abs(x) \to x \Leftarrow x \geq 0 \to^* true$$
$$abs(x) \to -x \Leftarrow x < 0 \to^* true$$

The symbol abs *is the root symbol of both conditional rules. We therefore replace every occurrence of the unary function* abs *in* $\mathcal{R}$ *by a new ternary function* abs' *that holds two additional conditional arguments. In the first conditional argument we encode the condition of the first conditional rule, in the second conditional argument the one of the second conditional rule. It is therefore possible to evaluate multiple conditions in parallel. Indeed the main motivation for the transformation is to apply parallel narrowing to functional logic programs that are represented by a CTRSs.*

An uninitialized conditional argument contains the special constant $\bot$*. This constant indicates that we can can insert the corresponding condition.*

The transformation returns the following unconditional rules

$$abs'(x, \bot, \bot) \to abs'(x, x \geq 0, x < 0)$$
$$abs'(x, true, y_2) \to x$$
$$abs'(x, y_1, true) \to -x$$

The transformation of [1] is only suitable for overlay CTRSs because overlapping rewrite steps might leave "garbage" terms from outdated evaluation attempts that can cause unsoundness:

Example 10 ([1, Example 6])**.** *Consider the following non-overlay CTRS and its transformation:*

$$\mathcal{R} = \left\{ \begin{array}{l} f(g(x)) \to x \Leftarrow x \to^* s(0) \\ g(s(x)) \to g(x) \end{array} \right\} \qquad \mathbb{T}_{[1]}(\mathcal{R}) = \left\{ \begin{array}{r} f'(g(x), \bot) \to f'(g(x), x) \\ f'(g(x), s(0)) \to x \\ g(s(x)) \to g(x) \end{array} \right\}$$

In the CTRS $\mathcal{R}$, $f(g(s(0)))$ rewrites to $s(0)$ and to $f(g(0))$. The latter term is irreducible because the condition $0 \to^* s(0)$ is not satisfiable. Nevertheless we obtain the following unsound derivation in $\mathbb{T}_{[1]}(\mathcal{R})$:

$$f'(g(s(0)), \bot) \to f'(g(s(0)), s(0)) \to f'(g(0), s(0)) \to 0$$

To overcome this limitation for non-overlay systems, some extensions have been presented in the past.

In [30] and [32], the transformation is extended by an additional circumfix symbol $\{\ \}$ that adds layers around contracted redexes. This additional symbol is syntactically complex. The CTRS of Example 10 is transformed into the following unconditional rules:

$$\mathbb{T}_{[32]}(\mathcal{R}) = \left\{ \begin{array}{r} f'(g(x), \bot) \to f'(g(x), \{x\}) \\ f'(g(x), \{s(0)\}) \to \{x\} \\ g(s(x)) \to \{g(x)\} \\ f'(\{x\}, y) \to \{f'(x, \bot)\} \\ g(\{x\}) \to \{g(x)\} \\ s(\{x\}) \to \{s(x)\} \\ \{\{x\}\} \to \{x\} \end{array} \right\}$$

We do not obtain the unsound derivation anymore:

$$\begin{aligned} f'(g(s(0)), \bot) &\to f'(g(s(0)), \{s(0)\}) \to f'(\{g(0)\}, \{s(0)\}) \\ &\to \{f'(g(0), \bot)\} \to \{f'(g(0), \{0\})\} \end{aligned}$$

where the last term is irreducible.

In [6] we introduced another transformation that encodes the conditional arguments in function symbols that overlap with other rules. We then encode the condition not in f but in g instead and obtain the following transformed TRS:

$$\mathbb{T}_{[6]}(\mathcal{R}) = \left\{ \begin{array}{r} f(g'(x, \bot)) \to f(g'(x, x)) \\ f(g'(x, s(0))) \to x \\ g'(s(x), y) \to g'(x, \bot) \end{array} \right\}$$

$$f(g'(s(0),\bot)) \to f(g'(s(0),s(0))) \to f(g'(0,\bot)) \to f(g'(0,0))$$

Similar to [32] we do not obtain unsoundness anymore.

The transformation of [32] is *computationally equivalent* for many confluent CTRSs, i.e., every normalizing rewrite sequence in the transformed TRS corresponds to a normalizing rewrite sequence in the original CTRS. The transformation of [6] also yields computational equivalence for some non-overlay CTRSs, yet although it has syntactically better properties than [32], the latter is computational equivalent for more CTRSs.

In this thesis we will focus on unravelings because of their simple nature. We will also provide examples that the unraveling of [29] is sound for certain CTRSs for which other transformations are not sound (see e.g. Example 42 or Example 43).

3.2 Unravelings

Unravelings represent the simplest class of transformations of CTRSs into TRSs. They stem from [14] and have been refined for various purposes ([15,24,29]). Unravelings differ from other transformations mainly by preserving the original signature which enables them to keep unconditional original rules. Other transformations like e.g. [1] require a transformation of terms.

The main idea of simulating conditional rewrite sequences using unravelings is as follows: An oriented conditional rule $\alpha : l \to r \Leftarrow s \to^* t$ is applicable to a term $l\sigma$ if there is a derivation $s\sigma \to^* t\sigma$. From this we derive the basic idea of transformations: If a term matches the left-hand side of a conditional rule l, then we wrap the left-hand side of the condition in some term. If after some rewrite steps this wrapped conditional argument rewrites into the right-hand side of the condition we reproduce the right-hand side of the conditional rule:

$$l \to r \Leftarrow s \to^* t \quad \Longrightarrow \quad \begin{cases} l \to C[s] & \text{introduction rule} \\ C[t] \to r & \text{elimination rule} \end{cases}$$

The conditional rule is therefore split into two unconditional rules. The rewrite step in which the *conditional argument* is introduced is the *introduction step*, the corresponding rule is the *introduction rule*. In the second rewrite step the conditional argument is eliminated, therefore the rule is the *elimination rule* and a rewrite step in which this rule is applied is an *elimination step*. If an elimination rule is applied to a term containing a conditional term, the term is *eliminated*.

For this approach it is necessary to introduce new function symbols to avoid overlaps with existing rules. Hence, the signature of the transformed system is a superset of the original signature $\mathcal{F}$. We denote the mapping in the following way: $\mathbb{U}(\mathcal{R}) = (\mathbb{U}(\mathcal{F}), \mathbb{U}(R))$.

We simulate a conditional rewrite step $l\sigma \to_{\mathcal{R}} r\sigma$ in unravelings in the following way:

$$l\sigma \to_{\mathbb{U}(\mathcal{R})} C[s\sigma] \to^*_{\mathbb{U}(\mathcal{R})} C[t\sigma] \to_{\mathbb{U}(\mathcal{R})} r\sigma$$

We will refer to terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$ (short $\mathcal{T}$) as *original terms*, the set of terms in the transformed system is $\mathcal{T}(\mathbb{U}(\mathcal{F}),\mathcal{V})$ (short $\mathbb{U}(\mathcal{T})$) and terms in $\mathcal{T}(\mathbb{U}(\mathcal{F}),\mathcal{V}) \setminus \mathcal{T}(\mathcal{F},\mathcal{V})$ are *mixed*

terms. In unravelings, new symbols are introduced to wrap the conditional argument(s). We call symbols in $\mathbb{U}(\mathcal{F}) \setminus \mathcal{F}$ *U-symbols*. Terms that are rooted by a U-symbol are *U-terms*.

3.2.1 Simultaneous Unraveling

In [14], two unravelings are defined, one for join-1-CTRSs $\mathbb{U}_{\mathsf{join}}$ and one for oriented 1-CTRSs $\mathbb{U}_{\mathsf{sim}}$. In the unraveling $\mathbb{U}_{\mathsf{sim}}$, we introduce a new function symbol U_α for each conditional rule α and then encode the conditions simultaneously in it.

Definition 1 (simultaneous unraveling $\mathbb{U}_{\mathsf{sim}}$ [14])**.** *Let α be the conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, ($k \geq 1$), then*

$$\mathbb{U}_{\mathsf{sim}}(\alpha) = \left\{ \begin{array}{ll} l \to U_\alpha(\overbrace{s_1, \ldots, s_k}^{\textit{conditions}}, \overbrace{\vec{X}}^{\textit{variables}}) & \textit{introduction rule} \\ U_\alpha(\underbrace{t_1, \ldots, t_k}_{\textit{conditions}}, \underbrace{\vec{X}}_{\textit{variables}}) \to r & \textit{elimination rule} \end{array} \right\}$$

where $X = \mathcal{V}ar(l)$.[1]

Unconditional rules remain unchanged: $\mathbb{U}_{\mathsf{sim}}(l \to r) = \{l \to r\}$.

The unraveled TRS $\mathbb{U}_{\mathsf{sim}}(\mathcal{R})$ *of some CTRS* $\mathcal{R}$ *is*

$$\mathbb{U}_{\mathsf{sim}}(\mathcal{R}) = \bigcup_{\alpha \in \mathcal{R}} \mathbb{U}_{\mathsf{sim}}(\alpha)$$

The unraveling for join systems corresponds to the unraveling $\mathbb{U}_{\mathsf{sim}}$ if we replace a join-condition $s \downarrow t$ by two oriented conditions $s \to^* z, t \to^* z$ where z is a new variable. Observe that in this case the unraveled TRS is not a 1-CTRS anymore, still $\mathbb{U}_{\mathsf{sim}}$ returns a valid TRS.

In this unraveling, every conditional rule is split into two unconditional rules. In the first rule, a term rooted by a U-symbol (a U-term) is introduced along with the left-hand sides of the conditions.

The right-hand sides of the rule and the conditions might contain variables that do not occur in the right-hand side of the conditions. In order to avoid extra variables in $\mathbb{U}(\mathcal{R})$, we need to preserve the variable bindings of the left-hand side of the introduction rule. Hence, we can partition the arguments of U-symbols into two groups: The conditions are inside conditional arguments while the variable bindings are inside *variable arguments*.

Conditions are verified in the following way: If a term matches the left-hand side of the conditional rule we rewrite it to the U-term containing all left-hand sides of the conditions. If the left-hand sides of the conditions rewrite to the corresponding right-hand sides, then we reproduce the right-hand side of the conditional rule, using the variable bindings stored in the variable arguments. The simulation of conditional rewrite steps using the simultaneous unraveling is illustrated in Figure 3.1.

[1] At some points in [14] r is used instead of l. Furthermore, in [14, Definition 6.1] all variables are encoded from left to right so that for non-left-linear conditional rules there are multiple variable arguments for the same variable.

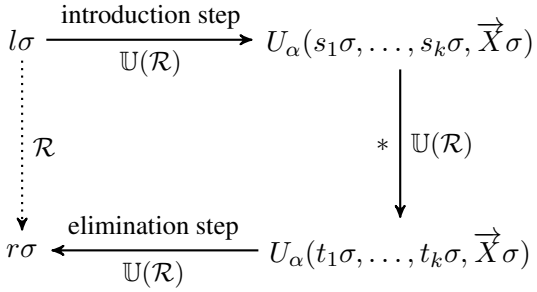

Figure 3.1: Simulation of conditional rewrite steps using simultaneous unraveling

Example 11. *Consider the normal 1-CTRS of Example 2:*

$$\mathcal{R} = \left\{ \begin{array}{rr} s(p(x)) \to x & pos(0) \to false \\ p(s(x)) \to x & pos(s(0)) \to true \\ pos(s(x)) \to true \Leftarrow pos(x) \to^* true & \\ pos(p(x)) \to false \Leftarrow pos(x) \to^* false & \end{array} \right\}$$

The conditional rules are unraveled in the following way:

$$\mathbb{U}_{\mathsf{sim}}(pos(s(x)) \to true \Leftarrow pos(x) \to^* true) = \left\{ \begin{array}{r} pos(s(x)) \to U_\alpha(pos(x), x) \\ U_\alpha(true, x) \to true \end{array} \right\}$$

$$\mathbb{U}_{\mathsf{sim}}(pos(p(x)) \to false \Leftarrow pos(x) \to^* false) = \left\{ \begin{array}{r} pos(p(x)) \to U_\beta(pos(x), x) \\ U_\beta(false, x) \to false \end{array} \right\}$$

We can simulate a conditional rewrite sequence starting from $pos(s(s(0)))$ in $\mathbb{U}_{\mathsf{sim}}(\mathcal{R})$:

$$pos(s(s(0))) \xrightarrow{(1)} U_\alpha(pos(s(0)), s(0)) \to U_\alpha(true, s(0)) \xrightarrow{(2)} true$$

The rewrite step (1) is the introduction step and (2) the elimination step.

Observe, that although $\mathcal{R}$ is confluent $\mathbb{U}_{\mathsf{sim}}(\mathcal{R})$ is not. For instance, $pos(p(s(0)))$ has at least two normalforms:

$$\begin{array}{l} pos(p(s(0))) \to pos(0) \to false \\ pos(p(s(0))) \to U_\beta(pos(s(0)), s(0)) \to U_\beta(true, s(0)) \end{array}$$

The simultaneous unraveling returns a valid TRS for oriented 1-CTRSs and also for deterministic CTRSs provided the left-hand sides of the conditions do not contain any extra variables. Check with the variable condition

3.2.2 Sequential Unraveling

The definition of deterministic CTRSs implies that in every conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ the variables in the conditions are arranged in the following way:

$$\mathcal{V}ar(s_i) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_{i-1}) \text{ for all } i \in \{1, \ldots, k\} \text{ and } \mathcal{V}ar(r) \subseteq \mathcal{V}ar(l, t_1, \ldots, t_k)$$

We therefore obtain all possible variable bindings for deterministic extra variables by sequentially evaluating the conditions. Hence, in the unraveling that is introduced in [15] the conditions are also evaluated in a sequential way. For this purpose it introduces one new U-symbol and one new unconditional rule for each condition.

The first argument in these sequential unravelings is the conditional argument while all other arguments contain variable arguments. Apart from introduction and elimination rules we obtain *switch rules* in which we switch to the next condition.

The following unraveling stems from [15] with a slight modification in [29].

Definition 2 (sequential unraveling $\mathbb{U}_{\mathsf{seq}}$ [29, Definition 7.2.33])**.** *Let α be the deterministic conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, then $\mathbb{U}_{\mathsf{seq}}(\alpha)$ is defined as follows:*

$$\mathbb{U}_{\mathsf{seq}}(\alpha) = \left\{ \begin{array}{rl} l \to U_1^\alpha(s_1, \overrightarrow{X_1}) & \textit{introduction rule} \\ U_1^\alpha(t_1, \overrightarrow{X_1}) \to U_2^\alpha(s_2, \overrightarrow{X_2}) & \textit{switch rule} \\ \vdots \quad \vdots \quad \vdots & \vdots \\ U_{k-1}^\alpha(t_{k-1}, \overrightarrow{X_{k-1}}) \to U_k^\alpha(s_k, \overrightarrow{X_k}) & \textit{switch rule} \\ U_k^\alpha(t_k, \overrightarrow{X_k}) \to r & \textit{elimination rule} \end{array} \right\}$$

where $X_i = \mathcal{V}ar(l, t_1, \ldots, t_{i-1})$.

Unconditional rules remain unchanged: $\mathbb{U}_{\mathsf{seq}}(l \to r) = \{l \to r\}$.

The unraveled TRS $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ of some CTRS is

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \bigcup_{\alpha \in \mathcal{R}} \mathbb{U}_{\mathsf{seq}}(\alpha)$$

Conditions are verified one by one. If a condition is satisfied we apply a switch rule and evaluate the next condition, until all conditions are satisfied. Figure 3.2 illustrates the verification of conditions in the sequential unraveling.

Example 12. *Consider the following deterministic CTRS:*

$$\mathcal{R} = \left\{ \begin{array}{l} x + 0 \to x \\ x + s(y) \to s(x + y) \\ quad(x) \to z \Leftarrow x + x \to^* y, y + y \to^* z \end{array} \right\}$$

Although $\mathcal{R}$ is a 3-CTRS all extra variables are deterministic. Therefore $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ returns an unconditional TRS without extra variables. Using the sequential unraveling, the conditional

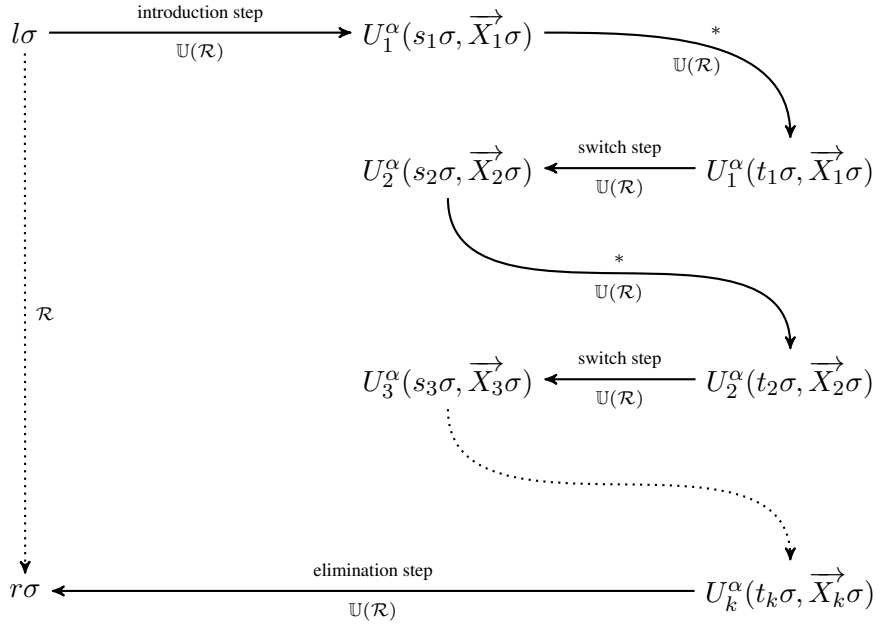

Figure 3.2: Simulation of conditional rewrite steps in the sequential unraveling

rule is unraveled into three unconditional rules:

$$\mathbb{U}_{\mathsf{seq}}(quad(x) \to z \Leftarrow x + x \to^* y, y + y \to^* z) = \left\{ \begin{array}{c} quad(x) \to U_1^\alpha(x + x, x) \\ U_1^\alpha(y, x) \to U_2^\alpha(y + y, x, y) \\ U_2^\alpha(z, x, y) \to z \end{array} \right\}$$

$quad(s(0))$ rewrites to $s(s(s(s(0))))$ in $\mathcal{R}$ because the condition $x + x \to^* y$ is satisfied for $x = s(0)$ and $y = s(s(0))$, and $y + y \to^* z$ for $z = s(s(s(s(0))))$.

In $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$, this corresponds to the following rewrite sequence:

$$\begin{aligned} quad(s(0)) &\to U_1^\alpha(s(0) + s(0), s(0)) \to^* U_1^\alpha(s(s(0)), s(0)) \\ &\to U_2^\alpha(s(s(0)) + s(s(0)), s(0), s(s(0))) \to^* U_2^\alpha(s(s(s(s(0)))), s(0), s(s(0))) \\ &\to s(s(s(s(0)))) \end{aligned}$$

Observe that we can simulate $\mathbb{U}_{\mathsf{sim}}$ using $\mathbb{U}_{\mathsf{seq}}$ by grouping the terms on the left-hand side and right-hand side of the conditions of the conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ into one term $l \to r \Leftarrow (s_1, \ldots, s_k) \to^* (t_1, \ldots, t_k)$ before applying the unraveling.

3.2.3 Sequential Optimized Unraveling

In $\mathbb{U}_{\mathsf{sim}}$ and $\mathbb{U}_{\mathsf{seq}}$ we encode all variable bindings that occur during the simulation of a conditional rewrite step. In [24], the variable arguments are optimized, meaning that only those variable bindings are stored that are still required in the remaining conditions in the right-hand side of

the conditional rule. For instance, in Example 12 all variables are immediately used in the condition and then never used again. We therefore do not need to store any variable arguments in the U-terms in order to obtain a TRS without extra variables.

We refer to this unraveling as the sequential optimized unraveling $\mathbb{U}_{\mathsf{opt}}$:

Definition 3 (sequential optimized unraveling $\mathbb{U}_{\mathsf{opt}}$ [17])**.** *Let α be the deterministic conditional rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, then $\mathbb{U}_{\mathsf{opt}}(\alpha)$ is defined in the following way:*

$$\mathbb{U}_{\mathsf{opt}}(\alpha) = \left\{ \begin{array}{rl} l \to U_1^\alpha(s_1, \overrightarrow{X_1}) & \textit{introduction rule} \\ U_1^\alpha(t_1, \overrightarrow{X_1}) \to U_2^\alpha(s_2, \overrightarrow{X_2}) & \textit{switch rule} \\ \vdots \quad \vdots \quad \vdots & \vdots \\ U_{k-1}^\alpha(t_{k-1}, \overrightarrow{X_{k-1}}) \to U_k^\alpha(s_k, \overrightarrow{X_k}) & \textit{switch rule} \\ U_k^\alpha(t_k, \overrightarrow{X_k}) \to r & \textit{elimination rule} \end{array} \right\}$$

where $X_i = \mathcal{V}ar(l, t_1, \ldots, t_{i-1}) \cap \mathcal{V}ar(t_i, s_{i+1}, t_{i+1}, \ldots, s_k, t_k, r)$.

For unconditional rules $\mathbb{U}_{\mathsf{opt}}(l \to r) = \{l \to r\}$.

The unraveled TRS $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ for some CTRS $\mathcal{R}$ is defined as

$$\mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \bigcup_{\alpha \in \mathcal{R}} \mathbb{U}_{\mathsf{opt}}(\alpha)$$

Unraveling the CTRS of Example 12 with $\mathbb{U}_{\mathsf{opt}}$ yields a simpler TRS than $\mathbb{U}_{\mathsf{seq}}$:

Example 13. *Consider the DCTRS of Example 12. Unraveling the conditional rule of Example 12 using $\mathbb{U}_{\mathsf{opt}}$ yields the following unconditional rules:*

$$\mathbb{U}_{\mathsf{opt}}(quad(x) \to z \Leftarrow x + x \to^* y, y + y \to^* z) = \left\{ \begin{array}{l} quad(x) \to U_1^\alpha(x + x) \\ U_1^\alpha(y) \to U_2^\alpha(y + y) \\ U_2^\alpha(z) \to z \end{array} \right\}$$

The U-terms in the unraveled TRS do not encode the variables x and y because they are not used in the right-hand side of the conditional rule or any of the following condition.

The rewrite sequence of Example 12 is simulated in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ in the following way:

$$\begin{aligned} quad(s(0)) &\to U_1^\alpha(s(0) + s(0)) \to^* U_1^\alpha(s(s(0))) \to U_2^\alpha(s(s(0)) + s(s(0))) \\ &\to U_2^\alpha(s(s(s(s(0))))) \to s(s(s(s(0)))) \end{aligned}$$

3.3 Ultra-Properties

Transformations allow us to simulate conditional rewrite sequences using unconditional rewrite steps. Furthermore, we can analyze properties of the transformed TRS and thereby obtain a better understanding of CTRSs. In [14], the notion of *ultra-properties* was introduced for this purpose:

Definition 4 (ultra-properties [14, Definition 3.2])**.** *A CTRS $\mathcal{R}$ has the ultra-property $\mathbb{U}$-$\mathcal{P}$ for some unraveling $\mathbb{U}$, if $\mathbb{U}(\mathcal{R})$ has the property $\mathcal{P}$.*

Ultra-properties can describe certain properties of CTRSs better than the original definition of these properties for CTRSs because they flatten the structure of conditional rules and therefore also consider properties that are hidden in the conditions. We can explain why certain CTRSs do not satisfy certain criteria of unconditional TRSs by analyzing the corresponding ultra-properties:

Example 14. *Example 7 presented a join-1-CTRS that is not confluent although it is orthogonal. The join CTRS of Example 7 corresponds to the following - also orthogonal - deterministic CTRS:*

$$\mathcal{R} = \left\{ \begin{array}{c} f(x) \to b \Leftarrow f(x) \to^* z, x \to^* z \\ a \to f(a) \end{array} \right\}$$

Unraveling the conditional rule gives rise to the following unconditional rules:

$$\mathbb{U}_{\mathsf{seq}}(f(x) \to b \Leftarrow f(x) \to^* z, x \to^* z) = \left\{ \begin{array}{r} f(x) \to U_1^\alpha(f(x), x) \\ U_1^\alpha(z, x) \to U_2^\alpha(x, x, z) \\ U_2^\alpha(z, x, z) \to b \end{array} \right\}$$

The unraveled TRS is still non-overlapping, yet it is not orthogonal because of the non-left-linear rule $U_2^\alpha(z, x, z) \to b$.

The notion of ultra-properties depends on the concrete unraveling that is used. We obtain different ultra-properties for different unravelings. By analyzing the definitions of the unravelings, we can show that $\mathbb{U}_{\mathsf{seq}}$-left-linearity and $\mathbb{U}_{\mathsf{opt}}$-left-linearity are equivalent, but for right-linearity and non-erasingness they differ:

Example 15. *Consider the conditional rule*

$$\alpha : f(x) \to g(y) \Leftarrow x \to^* y.$$

It is unraveled in the following way using $\mathbb{U}_{\mathsf{seq}}$ and $\mathbb{U}_{\mathsf{opt}}$:

$$\mathbb{U}_{\mathsf{seq}}(f(x) \to g(y) \Leftarrow x \to^* y) = \left\{ \begin{array}{r} f(x) \to U_\alpha^1(x, x) \\ U_\alpha^1(y, x) \to g(y) \end{array} \right\}$$

$$\mathbb{U}_{\mathsf{opt}}(f(x) \to g(y) \Leftarrow x \to^* y) = \left\{ \begin{array}{r} f(x) \to U_\alpha^1(x) \\ U_\alpha^1(y) \to g(y) \end{array} \right\}$$

The conditional rule is right-linear and erasing. Furthermore, it is $\mathbb{U}_{\mathsf{opt}}$-right-linear and $\mathbb{U}_{\mathsf{opt}}$-non-erasing, but not $\mathbb{U}_{\mathsf{seq}}$-right-linear and not $\mathbb{U}_{\mathsf{seq}}$-non-erasing, i.e., $\mathbb{U}_{\mathsf{seq}}$-erasing.

3.4 Term Translations

In derivations of an unraveled CTRS we obtain U-terms that represent intermediate evaluation steps in conditions. These terms correspond to a meta-state in the CTRS. For theoretical and also practical purposes it is preferable to be able to map such intermediate evaluations into terms in the original CTRS. For all original terms, such term mappings must be the identity of terms.

Backtranslation tb

In an ongoing derivation it is unknown, whether a conditional argument in some U-term is eventually satisfied. In such cases it is intuitive to replace the U-term by the corresponding left-hand side of the conditional rule. We can extract the variable arguments of the U-term and insert them into the left-hand side of the conditional rule. We refer to this mapping as the *backtranslation* tb:

Definition 5 (backtranslation)**.** *The mapping* $\mathsf{tb} : \mathbb{U}(\mathcal{T}) \to \mathcal{T}$ *is defined as*

$$\mathsf{tb}(t) = \begin{cases} x & \text{if } t = x \in \mathcal{V} \\ f(\mathsf{tb}(t_1), \ldots, \mathsf{tb}(t_{\mathsf{ar}(f)})) & \text{if } t = f(t_1, \ldots, t_{\mathsf{ar}(f)}) \text{ and } f \in \mathcal{F} \\ l\,\mathsf{tb}(\sigma) & \text{if } t = U_j^{\alpha}(u, \overrightarrow{X_j}\sigma) \end{cases}$$

where l *is the left-hand side of the rule* α*. We extend* tb *to substitutions in the following way:* $x\,\mathsf{tb}(\sigma) = \mathsf{tb}(x\sigma)$.

Observe that tb cannot always be defined in a satisfying way for $\mathbb{U}_{\mathsf{opt}}$ because $\mathbb{U}_{\mathsf{opt}}$ does not preserve all variable bindings of the left-hand side of a conditional rule:

Example 16. *Consider the conditional rule of Example 5 and its unraveled TRS using* $\mathbb{U}_{\mathsf{opt}}$*:*

$$\mathbb{U}_{\mathsf{opt}}(or(x,y) \to true \Leftarrow x \to^* true) = \left\{ \begin{array}{r} or(x,y) \to U_1^{\mathsf{opt},\alpha}(x) \\ U_1^{\mathsf{opt},\alpha}(true) \to true \end{array} \right\}$$

We obtain the derivation $or(false, false) \to U_1^{\mathsf{opt},\alpha}(false)$*. Backtranslating the U-term using* tb *yields the following term:*

$$\mathsf{tb}(U_1^{\mathsf{opt},\alpha}(false)) = or(x,y)\,\mathsf{tb}(\sigma) = or(x,y)$$

where σ *is the empty substitution because the U-term does not encode any variable arguments.*

Using $\mathbb{U}_{\mathsf{seq}}$ *we obtain the following rules:*

$$\mathbb{U}_{\mathsf{seq}}(or(x,y) \to true \Leftarrow x \to^* true) = \left\{ \begin{array}{r} or(x,y) \to U_1^{\mathsf{seq},\alpha}(x,x,y) \\ U_1^{\mathsf{seq},\alpha}(true,x,y) \to true \end{array} \right\}$$

$\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ *gives rise to the derivation* $or(false, false) \to U_1^{\mathsf{seq},\alpha}(false, false, false)$*. Backtranslating the U-term using* tb *now yields the following original term:*

$$\mathsf{tb}(U_1^{\mathsf{seq},\alpha}(false, false, false)) = or(x,y)\,\mathsf{tb}(\sigma) = or(false, false)$$

where $\sigma = \{x \mapsto false, y \mapsto false\}$.

Since $\mathbb{U}_{\mathsf{seq}}$ encodes all variables of the left-hand side in U-terms, $\mathtt{tb}$ is properly defined for all $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$.

Translation Forward $\mathtt{tf}$

If we know that the condition of a U-term is satisfied we can use the *translation forward* $\mathtt{tf}$ that replaces U-terms by the corresponding right-hand side of the conditional rule where we extract the variable arguments from the U-term similar to $\mathtt{tb}$. This will be in particular useful for non-erasing CTRSs:

Definition 6 (translation forward)**.** *The mapping* $\mathtt{tf} : \mathbb{U}(\mathcal{T}) \to \mathcal{T}$ *is defined as*

$$\mathtt{tf}(t) = \begin{cases} x & \text{if } t = x \in \mathcal{V} \\ f(\mathtt{tf}(t_1), \ldots, \mathtt{tf}(t_{\mathsf{ar}(f)})) & \text{if } t = f(t_1, \ldots, t_{\mathsf{ar}(f)}) \text{ and } f \in \mathcal{F} \\ r\,\mathtt{tf}(\sigma) & \text{if } t = U^{\alpha}_{j}(u, \overrightarrow{X_j}\sigma) \end{cases}$$

where r *is the right-hand side of the rule* α *and (by abuse of notation) we extend* $\mathtt{tf}$ *to substitutions in the following way:* $x\,\mathtt{tf}(\sigma) = \mathtt{tf}(x\sigma)$.

Similar to $\mathtt{tb}$, $\mathtt{tf}$ is not always properly defined. In particular 3-CTRSs contain extra variables in the right-hand side. These variables are not encoded in all corresponding U-terms. Therefore, there is no term to which these variable arguments could be mapped.

Example 17. *Consider the conditional rule of Example 12 that is unraveled into the following rules using* $\mathbb{U}_{\mathsf{opt}}$*:*

$$\mathbb{U}_{\mathsf{opt}}(quad(x) \to z \Leftarrow x + x \to^* y, y + y \to^* z) = \left\{ \begin{array}{r} quad(x) \to U^{\alpha}_{1}(x + x) \\ U^{\alpha}_{1}(y) \to U^{\alpha}_{2}(y + y) \\ U^{\alpha}_{2}(z) \to z \end{array} \right\}$$

Consider the rewrite step $quad(0) \to U^{\alpha}_{1}(0 + 0)$ *in* $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$*. Translating the contracted redex* $\mathtt{tf}(U^{\alpha}_{1}(0 + 0))$ *yields* z *because the U-term does not contain a variable argument* z*.*

$\mathtt{tf}$ is properly defined for $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ and $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ if $\mathcal{V}ar(r) \subseteq \mathcal{V}ar(l)$, i.e., for 2-CTRSs for $\mathbb{U}_{\mathsf{seq}}$ and $\mathbb{U}_{\mathsf{opt}}$.

CHAPTER 4

Soundness

4.1 Introduction

In order to prove properties like (operational) termination or confluence using unravelings we need to prove equivalence properties of the rewrite relation of the transformed TRS and the original CTRS. One major challenge is that $\mathbb{U}(\mathcal{R})$ has a richer signature than $\mathcal{R}$. In derivations in $\mathbb{U}(\mathcal{R})$ we not only obtain terms in $\mathcal{T}(\mathcal{F},\mathcal{V})$ but also terms with new symbols in $\mathbb{U}(\mathcal{F}) \setminus \mathcal{F}$. Therefore, the rewrite relation of the transformed TRS usually contains the rewrite relation of the CTRS. This direction is called completeness.

Soundness is usually not satisfied. In fact we obtain different soundness properties for different transformations. Nevertheless, we need soundness to prove properties like confluence or (operational) non-termination. The reason for unsoundness are terms that represent partially evaluated conditions, i.e., U-terms. In the transformed CTRS, U-terms may be "crossed" with other U-terms in rewrite steps using non-linear rules, and also may rewrite variable arguments inside U-terms.

In the past soundness properties of unravelings have been investigated several times:

- In [14] an example is presented that is a counterexample for soundness of $\mathbb{U}_{\mathsf{sim}}$. The normal 1-CTRS contains a sophisticated combination of non-linear rules. It is then shown that left-linearity is sufficient for soundness of normal 1-CTRSs for $\mathbb{U}_{\mathsf{sim}}$, where some flaws of the proof are corrected in [29].

- In [24] it is shown that a combination of context sensitive rewriting and rewriting using the membership condition is sufficient for soundness of the optimized unraveling $\mathbb{U}_{\mathsf{opt}}$ for deterministic CTRSs.

- In [31], it is proven that $\mathbb{U}_{\mathsf{seq}}$ is sound for context-sensitive rewriting using the replacement map $\mu(U_j^\alpha) = \{1\}$ for all U-symbols $U_j^\alpha \in \mathbb{U}(\mathcal{F}) \setminus \mathcal{F}$, and $\mu(f) = \{1,\ldots,\mathsf{ar}(f)\}$ for all original symbols $f \in \mathcal{F}$.

- In [25], it is shown that $\mathbb{U}_{\mathsf{opt}}$-left-linearity is sufficient for soundness of $\mathbb{U}_{\mathsf{opt}}$ for DCTRSs. This result also implies soundness for $\mathbb{U}_{\mathsf{opt}}$-right-linear and $\mathbb{U}_{\mathsf{opt}}$-non-erasing CTRSs. In [25] it is also shown that soundness for $\mathbb{U}_{\mathsf{opt}}$ implies soundness for $\mathbb{U}_{\mathsf{seq}}$. The reverse does not hold. Example 5 is in fact a counterexample.

- In [7] we improved the results of [14] by proving that we even obtain soundness for non-left-linear CTRSs in which all non-left-linear variables are erased. We call this property *weak left-linearity*. We also proved soundness for $\mathbb{U}_{\mathsf{sim}}$-non-erasing normal 1-CTRSs, $\mathbb{U}_{\mathsf{sim}}$-right-linear normal 1-CTRSs and confluent normal 1-CTRSs while providing counterexamples for many other syntactic criteria.

- In [8] we adapt some of our results in [7] to DCTRSs, yet the case of deterministic TRSs is significantly more complex. Hence, we obtained many negative results, like unsoundness for non-erasing DCTRSs and unsoundness for confluent DCTRSs. Nonetheless we proved weaker soundness results for these cases.

In this chapter we will investigate soundness properties and analyze which syntactic properties are sufficient for soundness. For this purpose we will in detail analyze possible sources for unsoundness, providing an in-depth-analysis of counterexamples. We will provide two approaches to prove soundness: First we define syntactic criteria for which we can prove soundness directly using the term translations $\mathtt{tb}$ and $\mathtt{tf}$. Since this approach is of only limited use we will define a rewrite strategy that implies soundness. We finally prove that every derivation in weakly left-linear and also weakly right-linear DCTRSs can be translated into such a U-eager rewrite sequence, thus proving soundness.

4.2 Soundness Properties

The terminology of soundness and completeness properties differs in the literature. Soundness is sometimes called simulation-completeness or simulation-soundness while complete unravelings are sometimes called simulation-preserving. In order to avoid confusion we introduced a unified framework for transformations of CTRSs into TRSs that also covers unravelings in [6]. We therefore follow the notions of this paper where, for the sake of readability, we skip the mapping of terms that is required for transformations that are not unravelings.

In order to show that we actually can simulate conditional rewrite sequences in $\mathbb{U}(\mathcal{R})$ we define *completeness for reduction* or simply *completeness*.

Definition 7 (complete w.r.t. reducibility [6])**.** *An unraveling $\mathbb{U}$ is complete w.r.t. reducibility (or simply complete) for a CTRS $\mathcal{R}$ if for every derivation $u \to^*_{\mathcal{R}} v$ such that $u, v \in \mathcal{T}$ also $u \to^*_{\mathbb{U}(\mathcal{R})} v$.*

In [14, Theorem 4.2] it is shown that $\mathbb{U}_{\mathsf{sim}}$ is complete. The proof itself can be adapted to the unravelings $\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$.

Completeness is important to prove termination properties because using completeness we can prove that an infinite derivation in the original CTRS must translate into an infinite derivation

in the transformed system. This plays an important role to prove operational termination of the original CTRS by termination of the unraveled TRS.

For transformations we usually obtain completeness because transformations increase the power of the rewrite relation by adding new symbols, yet they do not decrease it.

The other direction, that $u \to^*_{\mathbb{U}(\mathcal{R})} v$ implies $u \to^*_{\mathcal{R}} v$ for all $u, v \in \mathbb{U}(\mathcal{T})$ is usually not satisfied because we cannot rewrite mixed terms in $\mathcal{R}$. We therefore restrict this notion to original terms, and obtain *soundness w.r.t. reducibility*:

Definition 8 (soundness w.r.t. reducibility [6])**.** *An unraveling $\mathbb{U}$ is sound w.r.t. reducibility (or simply sound) for a CTRS $\mathcal{R}$ if for all $u, v \in \mathcal{T}$ such that $u \to^*_{\mathbb{U}(\mathcal{R})} v$ also $u \to^*_{\mathcal{R}} v$.*

The notions of soundness and completeness are not symmetric because in both notions we only consider original terms. Mixed terms represent an unfinished attempt to verify some condition, hence we do not consider them in the original CTRS.

Nonetheless, we obtain such terms in derivations and they might reveal some unexpected behaviour. We therefore do not obtain soundness for reductions in general.

Another soundness property that we are interested in is soundness w.r.t. joinability:

Definition 9 (soundness w.r.t. joinability)**.** *An unraveling $\mathbb{U}$ is sound w.r.t. joinability for a CTRS $\mathcal{R}$ if for all $u, v \in \mathcal{T}$ such that $u \downarrow_{\mathbb{U}(\mathcal{R})} v$ then $u \downarrow_{\mathcal{R}} v$.*

Soundness w.r.t. joinability is important to prove confluence of CTRSs using unravelings because we can prove that two terms are joinable in the conditional rewrite system by proving that they are joinable in the transformed TRS.

Our original notion of soundness requires that the last term in a reduction is a term without U-terms. Nonetheless, we can use the backtranslation to translate U-terms that occur in the last term of a reduction and thereby obtain a more strict soundness property.

Definition 10 (soundness w.r.t. the backtranslation tb)**.** *An unraveling $\mathbb{U}$ is sound w.r.t. tb for a CTRS $\mathcal{R}$ if for all $u \in \mathcal{T}$ such that $u \to^*_{\mathbb{U}(\mathcal{R})} v$ also $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$.*

Soundness w.r.t. tb is stronger than soundness, and since tb is total for all terms in the transformed system we can use it to prove further soundness properties. There is the following important connection between soundness w.r.t. tb and soundness for joinability:

Lemma 1. *If an unraveling $\mathbb{U}$ is sound w.r.t. tb for a CTRS $\mathcal{R}$, then it is also sound w.r.t. joinability.*

Proof. Let $u, v \in \mathcal{T}$ be two terms such that $u \downarrow_{\mathbb{U}(\mathcal{R})} v$. This implies that there is a $w \in \mathbb{U}(\mathcal{F})$ such that $u \to^*_{\mathbb{U}(\mathcal{R})} w$ and $v \to^*_{\mathbb{U}(\mathcal{R})} w$. If $\mathbb{U}$ is sound w.r.t. tb then therefore $u \to^*_{\mathcal{R}} \mathsf{tb}(w)$ and $v \to^*_{\mathcal{R}} \mathsf{tb}(w)$. Therefore, $u \downarrow_{\mathcal{R}} v$. □

We will show in Example 23 that the reverse of this result does not hold.

Observe that soundness w.r.t. tb does not imply $\mathsf{tb}(u) \to^*_{\mathcal{R}} \mathsf{tb}(v)$ for all $u, v \in \mathbb{U}(\mathcal{T})$ such that $u \to^*_{\mathbb{U}(\mathcal{R})} v$. Example 6 is a counterexample.

4.3 Sortedness

We encode variable bindings and conditional arguments in U-terms. Therefore, one source of unsoundness are rewrite steps in variable bindings that modify the status of satisfiability of the conditional argument.

Example 18. *Consider the following CTRS and its unravelled corresponding system (the unravelings $\mathbb{U}_{\mathsf{sim}}$, $\mathbb{U}_{\mathsf{seq}}$ and $\mathbb{U}_{\mathsf{opt}}$ are equivalent in this case):*

$$\mathcal{R} = \left\{ \begin{array}{l} f(x) \to x \Leftarrow s(x) \to^* t(x) \\ s(a) \to t(b) \\ a \to b \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{r} f(x) \to U_1^\alpha(s(x), x) \\ U_1^\alpha(t(x), x) \to x \\ s(a) \to t(b) \\ a \to b \end{array} \right\}$$

The unraveled TRS gives rise to the following derivation:

$$f(a) \to U_1^\alpha(s(a), a) \Vdash U_1^\alpha(t(b), b) \to b$$

In $\mathcal{R}$, $f(a)$ does not rewrite to b. In fact, we cannot apply the conditional rule to any term because the condition $s(x) \to^* t(x)$ is unsatisfiable.

Observe that by adding one rule we obtain the CTRS $\mathcal{R} \cup \{t(b) \to s(b)\}$ which is confluent. Still, we obtain unsoundness by the same argument as above.

In the unsound derivation of Example 18, the conditional argument rewrites to a matching right-hand side of the condition although the condition is unsatisfiable. Still it seems that the condition is satisfied because of rewrite steps in variable arguments that intercept rewrite steps in the conditional arguments.

We obtain derivations like in Example 18 only if certain conditions are met. Let $\alpha : l \to r \Leftarrow s \to^* t$ be some conditional rewrite rule. We might obtain unsoundness if there are two substitutions σ and τ such that $x\sigma \to^* x\tau$ for all $x \in \mathcal{V}ar(\alpha)$, and $s\sigma \to^*_{\mathcal{R}} t\tau$. In this case we obtain the following derivation in $\mathbb{U}(\mathcal{R})$:

$$l\sigma \to_{\mathbb{U}(\mathcal{R})} U_1^\alpha(s\sigma, \vec{X}\sigma) \to^*_{\mathbb{U}(\mathcal{R})} U_1^\alpha(t\tau, \vec{X}\tau) \to r\tau$$

This derivation is unsound if $l\sigma \not\to^*_{\mathcal{R}} r\tau$. Since $l\sigma \to^*_{\mathcal{R}} l\tau$ and $r\sigma \to^*_{\mathcal{R}} r\tau$ this is equivalent to the requirement that $l\sigma \not\to^*_{\mathcal{R}} r\sigma$ and $l\tau \not\to^*_{\mathcal{R}} r\tau$, i.e., that the conditions are not satisfied for neither σ nor τ.

Figure 4.1 illustrates all possible derivations in $\mathcal{R}$ in the conditions.

In order to avoid unsoundness in this case we need to restrict the distribution of variables in the conditions. We need to make sure that if $s\sigma \to^* t\tau$ and $x\sigma \to^* x\tau$, then also $s\sigma \to^* t\sigma$ or $s\tau \to^* t\tau$.

In many practically relevant cases this is trivially satisfied. For instance, for normal 1-CTRSs t is a ground term so that $t\tau = t\sigma = t$.

In general, it is enough to ensure that the left-hand side of a condition and its right-hand side do not share variables. We call this property *sortedness* because it depends on the order of variables in conditions and, as we will show later, also on the order of conditions.

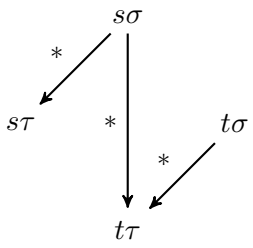

Figure 4.1: Derivations in non-sorted conditions

For the terms in Figure 4.1, sortedness implies $\mathcal{V}ar(s) \cap \mathcal{V}ar(t) = \emptyset$. In this case, $s\sigma = s\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} = s\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)})$ and $t\tau = t\tau\!\!\upharpoonright_{\mathcal{V}ar(t)} = t\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)})$. The rewrite sequence $s\sigma \to^* t\tau$ is therefore equivalent to $s\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)}) \to^* t\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)})$.

Hence, the condition is satisfied for the matcher $(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)})$ in $\mathcal{R}$ such that we obtain the following rewrite sequence in $\mathcal{R}$:

$$l\sigma \to^* l\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)}) \to r\,(\sigma\!\!\upharpoonright_{\mathcal{V}ar(s)} \circ \tau\!\!\upharpoonright_{\mathcal{V}ar(t)}) \to^* r\tau$$

Therefore, the rewrite sequence $l\sigma \to^*_{\mathbb{U}(\mathcal{R})} r\tau$ is sound.

For general deterministic CTRSs sortedness is defined as follows:

Definition 11 (sortedness)**.** *A CTRS is sorted if for all rules $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k \in \mathcal{R}$, $\mathcal{V}ar(s_1, \ldots, s_i) \cap \mathcal{V}ar(t_i, \ldots, t_k) = \emptyset$ for all $i \in \{1, \ldots, k\}$.*

The following example shows that sortedness and therefore soundness depends on the order of conditions:

Example 19. *Consider the following CTRS and its unraveling ($\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$ return the same TRS).*

$$\mathcal{R} = \left\{ \begin{array}{c} a \nearrow c \\ \quad\; \searrow d \\ f(x) \to x \Leftarrow x \to^* c, d \to^* x \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{c} a \nearrow c \\ \quad\; \searrow d \\ f(x) \to U_1^\alpha(x, x) \\ U_1^\alpha(c, x) \to U_2^\alpha(d, x) \\ U_2^\alpha(x, x) \to x \end{array} \right\}$$

The condition of the conditional rule is unsatisfiable: The only term that satisfies the second condition $d \to^* x$ is d itself, yet the first condition $d \to^* c$ is not satisfiable.

The CTRS is not sorted because $\mathcal{V}ar(s_1) \cap \mathcal{V}ar(t_2) = \{x\}$. In the transformed DCTRS we obtain the following unsound derivation:

$$f(a) \to U_1^\alpha(a, a) \to U_1^\alpha(c, a) \to U_2^\alpha(d, a) \to U_2^\alpha(d, d) \to d$$

If we transform the conditional rule using the simultaneous unraveling $\mathbb{U}_{\mathsf{sim}}$ we obtain the rules:

$$\mathbb{U}_{\mathsf{sim}}(f(x) \to x \Leftarrow x \to^* c, d \to^* x) \;=\; \left\{ \begin{array}{c} f(x) \to U_\alpha(x, d, x) \\ U_\alpha(c, x, x) \to x \end{array} \right\}$$

We obtain a similar unsound derivation:

$$f(a) \to U_\alpha(a,d,a) \to U_\alpha(c,d,a) \to U_\alpha(c,d,d) \to d$$

We can convert $\mathcal{R}$ into a sorted CTRS by switching positions of the conditions:

$$\mathbb{U}(f(x) \to x \Leftarrow d \to^* x, x \to^* c) \;=\; \left\{ \begin{array}{r} f(x) \to U_1^\alpha(d,x) \\ U_1^\alpha(x,x) \to U_2^\alpha(x,x) \\ U_2^\alpha(c,x) \to x \end{array} \right\}$$

Now, in order to apply the rule $U_1^\alpha(x,x) \to U_2^\alpha(x,x)$, x must be bound to some successor of d, yet the only candidate is d itself. Since d does not rewrite to c we cannot apply the elimination rule. Therefore, we do not obtain the unsound derivation anymore.

4.4 Negative Results

Sortedness implies that a satisfied conditional argument in the transformed TRS actually corresponds to a satisfiable condition in the original CTRS if the unraveling is sound. Since the rewrite sequence that we can extract from the conditional argument is shorter than the original rewrite sequence itself this gives us the possibility to prove soundness using induction on the length of derivations. We will therefore prove soundness for sorted CTRSs (or some stricter property).

4.4.1 Non-Linear CTRSs

In Example 18 and Example 19 we obtain unsoundness for non-sorted CTRSs. Sortedness itself is not sufficient to obtain soundness. We showed unsoundness of $\mathbb{U}_{\mathsf{opt}}$ for the (sorted) CTRS in Example 5.

For the unraveling $\mathbb{U}_{\mathsf{sim}}$ (and therefore also $\mathbb{U}_{\mathsf{seq}}$) [14] presents an unsound normal 1-CTRS. Since the right-hand sides of the conditions are ground terms this CTRS is also sorted according to our definition. In the following we will analyze unsoundness examples to give a better intuition for sufficient criteria for soundness.

The following example is a slight simplification of this CTRS:

Example 20 (Unsoundness-Example of [14])**.** *Consider the following CTRS and its unraveling (that is equivalent for $\mathbb{U}_{\mathsf{seq}}$, $\mathbb{U}_{\mathsf{opt}}$ and also for $\mathbb{U}_{\mathsf{sim}}$):*

$$\mathcal{R} = \left\{ \begin{array}{r} \begin{array}{ccc} a & \to & c \to e \\ & \searrow\!\!\nearrow & \\ b & \to & d \to k \end{array} \\ f(x) \to x \Leftarrow x \to^* e \\ g(x,x) \to h(x,x,f(k)) \\ h(d,x,x) \to A \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{r} \begin{array}{ccc} a & \to & c \to e \\ & \searrow\!\!\nearrow & \\ b & \to & d \to k \end{array} \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(e,x) \to x \\ g(x,x) \to h(x,x,f(k)) \\ h(d,x,x) \to A \end{array} \right\}$$

We obtain the following derivation in $\mathbb{U}(\mathcal{R})$*:*

$$\begin{aligned} g(f(a), f(b)) &\to^* g(U_1^\alpha(a,a), U_1^\alpha(b,b)) \to^* g(U_1^\alpha(c,d), U_1^\alpha(c,d)) \\ &\to h(U_1^\alpha(c,d), U_1^\alpha(c,d), f(k)) \to^* h(d, U_1^\alpha(c,d), f(k)) \\ &\to^* h(d, U_1^\alpha(k,k), U_1^\alpha(k,k)) \to A \end{aligned}$$

Yet, in $\mathcal{R}$ *there is no such derivation in* $\mathcal{R}$*: We can apply the non-linear rule* $g(x,x) \to h(x,x,f(k))$ *only if* x *is bound to a common reduct of* $f(a)$ *and* $f(b)$*. In order to apply the rule* $h(d,x,x) \to A$ *this common reduct must rewrite to* d*. Additionally, this common reduct must have a common reduct with* $f(k)$*. In* $\mathbb{U}(\mathcal{R})$*, the terms* $U_1^\alpha(c,d)$ *and* $U_1^\alpha(k,k)$ *satisfy this condition. The corresponding derivations are shown in Figure 4.2.*

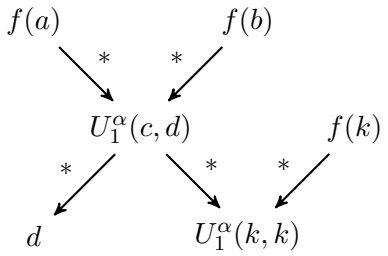

Figure 4.2: Derivations leading to unsoundness in Example 20

Yet, in $\mathcal{R}$ *there are no such terms. Figure 4.3 shows all common reducts of* $f(a)$ *and* $f(b)$ *in* $\mathcal{R}$ *and all rewrite steps between them. It is easy to see that there is no term that rewrites to* d *and has a common reduct with* $f(k)$*.*

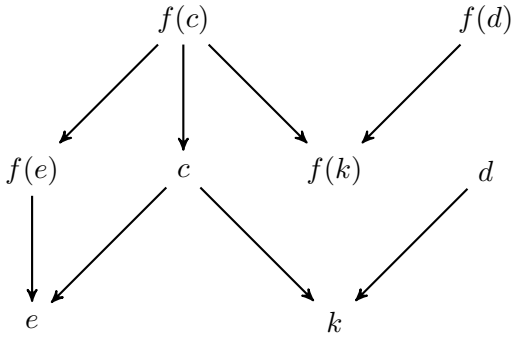

Figure 4.3: Common reducts of $f(a)$ and $f(b)$ in $\mathcal{R}$ in Example 20

Observe that this example is also a counterexample for soundness of joinability because $g(f(a), f(b)) \downarrow_{\mathbb{U}(\mathcal{R})} A$ *but* $g(f(a), f(b)) \not\downarrow_{\mathcal{R}} A$*.*

In order to better understand how non-linear rules give rise to unsoundness we incrementally construct a similar counterexample for all soundness properties that we defined above.

Example 21 (new unsoundness example)**.** *Consider the following CTRS and its unraveling:*

$$\mathcal{R}_1 = \left\{ \begin{array}{c} a \nearrow c \\ \quad \searrow d \\ f(x) \to x \Leftarrow x \to^* c \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}_1) = \left\{ \begin{array}{c} a \nearrow c \\ \quad \searrow d \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(c,x) \to x \end{array} \right\}$$

The unraveled TRS yields indeed soundness, even soundness w.r.t. tb*. Nonetheless, using* tb *to translate derivations gives rise to problems as we show using the following derivation:*

$$f(a) \to U_1^\alpha(a,a) \to^* U_1^\alpha(c,d) \to d$$

The variable argument in the U-term $U_1^\alpha(c,d)$ for x is d. Hence, although this derivation is sound w.r.t. tb *this derivation contradicts a similar property: We cannot backtranslate all single rewrite steps using* tb *because* $\mathsf{tb}(U_1^\alpha(c,d)) = f(d)$ *and* $\mathsf{tb}(d) = d$*, yet $f(d) \not\to^*_{\mathcal{R}_1} d$. We can exploit this property and develop a CTRS for which we do not obtain soundness w.r.t.* tb*:*

Consider the CTRS $\mathcal{R}_1 \cup \mathcal{R}_2$ where $\mathcal{R}_2$ is defined as follows:

$$\mathcal{R}_2 = \left\{ \begin{array}{c} b \nearrow c \\ \quad \searrow d \\ g(x,x) \to h(x,x) \end{array} \right\}$$

By a similar argument as above, the following derivation in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2)$

$$f(b) \to U_1^\alpha(b,b) \to^* U_1^\alpha(c,d) \to d$$

is also sound for tb *because $f(b) \to_{\mathcal{R}_1 \cup \mathcal{R}_2} b \to_{\mathcal{R}_1 \cup \mathcal{R}_2} d$, although the backtranslation of the rewrite step $U_1^\alpha(c,d) \to d$ does not return a valid derivation in the original CTRS:*

$$f(d) \not\to^*_{\mathcal{R}_1 \cup \mathcal{R}_2} d$$

Now consider the following derivation in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2)$:

$$\begin{aligned} g(f(a),f(b)) &\to^* g(U_1^\alpha(a,a),U_1^\alpha(b,b)) \to^* g(U_1^\alpha(c,d),U_1^\alpha(c,d)) \\ &\to h(U_1^\alpha(c,d),U_1^\alpha(c,d)) \to h(d,U_1^\alpha(c,d)) \to h(d,d) \end{aligned}$$

If we skip the last rewrite step we obtain a derivation that is not sound for tb*: In $\mathcal{R}_1 \cup \mathcal{R}_2$, the only term that is a common reduct of $f(a)$ and $f(b)$ that rewrites to d is d itself. In the unraveling there is also the term $U_1^\alpha(c,d)$ that can be reduced to d in a later rewrite step. Since* $\mathsf{tb}(U_1^\alpha(c,d)) = f(d)$*, the derivation*

$$g(f(a),f(b)) \to^* g(U_1^\alpha(c,d),U_1^\alpha(c,d)) \to h(U_1^\alpha(c,d),U_1^\alpha(c,d)) \to h(d,U_1^\alpha(c,d))$$

is unsound for tb *because $g(f(a),f(b))$ does not rewrite to $h(d,f(d))$ in $\mathcal{R}$.*

Next we want to develop a counterexample for soundness w.r.t. joinability. Since the terms $f(d)$ and $U_1^\alpha(c,d)$ are not joinable in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2)$ this derivation is not a counterexample for soundness w.r.t. joinability.

In order to force these two terms to become joinable in the unraveling we introduce two new constants k, m using the following rewrite rules:

$$\mathcal{R}_3 = \left\{ \begin{array}{l} c \to k \\ d \to k \\ \quad\; \to m \end{array} \right\}$$

Now, $f(d)$ and $U_1^\alpha(c,d)$ are joinable in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)$:

$$f(d) \to U_1^\alpha(d,d) \Vdash U_1^\alpha(k,m) \Vdash\!\!\leftarrow U_1^\alpha(c,d)$$

Furthermore, $U_1^\alpha(c,d)$ is a common reduct of $f(a)$ and $f(b)$ that is joinable with d.

Therefore, $g(f(a),f(b))$ and $h(d,f(d))$ are joinable in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)$. Yet $f(c)$ is a common reduct of $f(a), f(b)$ that is joinable with d ($f(c) \to^* k \;{}^*\!\leftarrow d$) in $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$. Nonetheless, $U_1^\alpha(c,d)$ is also joinable with m ($U_1^\alpha(c,d) \to d \to m$) but $f(c) \not\downarrow m$.

Therefore, the successor $h(m,f(k))$ of $h(d,f(d))$ is not joinable with $g(f(a),f(b))$ in $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$ but it is in $\mathbb{U}(\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3)$:

$$g(f(a),f(b)) \to^* h(d,U_1^\alpha(c,d)) \to^* h(m,U_1^\alpha(k,k)) \leftarrow^* h(m,f(k))$$

Hence the unravelings are not sound w.r.t. joinability for $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$. Observe that $\mathcal{R}_1 \cup \mathcal{R}_2 \cup \mathcal{R}_3$ is ultra-non-erasing.

By adding the rule $eq(x,x) \to true$ we then obtain unsoundness as in Example 5.

Since the right-hand side of the conditional rule is not an irreducible ground term anymore, we also add a rule $c \to e$ and update the conditional rule accordingly. In total, we obtain the following CTRS and its unraveling:

$$\mathcal{R} = \left\{ \begin{array}{l} a \to c \to e \\ \quad\;\searrow \\ b \to d \to k \\ \quad\;\searrow \\ \quad\; \to m \\ \\ f(x) \to x \Leftarrow x \to^* e \\ g(x,x) \to h(x,x) \\ eq(x,x) \to true \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{l} a \to c \to e \\ \quad\;\searrow \\ b \to d \to k \\ \quad\;\searrow \\ \quad\; \to m \\ \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(e,x) \to x \\ g(x,x) \to h(x,x) \\ eq(x,x) \to true \end{array} \right\}$$

The following derivation in $\mathbb{U}(\mathcal{R})$ is unsound for the reasons that we stated above:

$$\begin{aligned} eq(g(f(a),f(b)),h(m,f(k))) &\to^* eq(g(U_1^\alpha(c,d),U_1^\alpha(c,d)),h(m,f(k))) \\ &\to^* eq(h(U_1^\alpha(c,d),U_1^\alpha(c,d)),h(m,f(k))) \\ &\to^* eq(h(d,U_1^\alpha(c,d)),h(m,f(k))) \\ &\to^* eq(h(m,U_1^\alpha(k,k)),h(m,U_1^\alpha(k,k))) \to true \end{aligned}$$

By modifying the example, especially simplifying the unsound derivation, we obtain the CTRS of Example 20.

In the previous example we see that since the conditional arguments and variable arguments might become incompatible a sorted CTRS must satisfy several syntactical properties so that we obtain unsoundness. The unsound CTRS is non-left-linear and non-right-linear, furthermore it is erasing (also for the corresponding ultra-properties).

Without the erasing rule $eq(x,x) \to true$, we cannot remove U-terms in a derivation if the condition is not satisfied. Although we do not obtain soundness for non-erasing systems in general we will later prove soundness of certain non-erasing CTRSs, based on this observation.

4.4.2 Non-Erasing CTRSs

U-terms are used to simulate conditional rewrite steps. In Example 5 we have shown that U-terms are potentially dangerous if they represent an unfinished evaluation because they might have different properties than original terms. In rewrite sequences in ultra-non-erasing CTRSs that start and end in an original term all U-terms are eliminated and therefore the conditional argument was successfully verified. Hence, ultra-non-erasing CTRSs are one possible candidate to obtain soundness.

In fact, we have proven soundness for ultra-non-erasing normal 1-CTRSs for $\mathbb{U}_{\mathsf{sim}}$ in [7], although we do not obtain soundness w.r.t. $\mathtt{tb}$ (and therefore neither soundness w.r.t. joinability) (see Example 21). Yet, for DCTRSs we do not obtain soundness. We presented the following counterexample in [8]:

Example 22 ([8, Example 3])**.** *Consider the following DCTRS and its unraveling (again $\mathbb{U}_{\mathsf{seq}}$ and $\mathbb{U}_{\mathsf{opt}}$ are equivalent)*

$$\mathcal{R} = \left\{ \begin{array}{r} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ s(c) \nearrow t(k) \\ \quad \searrow t(l) \\ g(x,x) \to h(x,x) \\ f(x) \to \langle x,y\rangle \Leftarrow s(x) \to^* t(y) \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{r} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ s(c) \nearrow t(k) \\ \quad \searrow t(l) \\ g(x,x) \to h(x,x) \\ f(x) \to U_1^\alpha(s(x),x) \\ U_1^\alpha(t(y),x) \to \langle x,y\rangle \end{array} \right\}$$

$f(a)$ and $f(b)$ have the common reducts $\langle d,k\rangle$ and $\langle d,l\rangle$, yet, there is no term u that is a common reduct of $f(a)$ and $f(b)$ that rewrites to $\langle d,k\rangle$ and $\langle d,l\rangle$ (see Figure 4.4).

In the unraveling, $f(a)$ and $f(b)$ rewrite to $U_1^\alpha(s(c),d)$. This term also rewrites to $\langle d,k\rangle$ and $\langle d,l\rangle$. Figure 4.5 shows the rewrite steps.

In order to apply the non-linear rule $g(x,x) \to h(x,x)$ in $\mathcal{R}$ we need some common reduct of $f(a)$ and $f(b)$ that rewrites to $\langle d,k\rangle$ and $\langle d,l\rangle$. As we have shown in Figure 4.4, there is no such term. We therefore obtain the following unsound derivation in $\mathbb{U}(\mathcal{R})$:

$$\begin{aligned} g(f(a),f(b)) &\Vdash g(U_1^\alpha(s(a),a),U_1^\alpha(s(b),b)) \Vdash g(U_1^\alpha(s(c),a),U_1^\alpha(s(c),b)) \\ &\Vdash g(U_1^\alpha(s(c),d),U_1^\alpha(s(c),d)) \to h(U_1^\alpha(s(c),d),U_1^\alpha(s(c),d)) \\ &\Vdash h(U_1^\alpha(t(k),d),U_1^\alpha(t(l),d)) \Vdash h(\langle d,k\rangle,\langle d,l\rangle) \end{aligned}$$

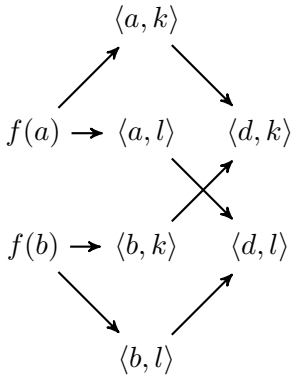

Figure 4.4: Rewrite steps in $\mathcal{R}$ in Example 22

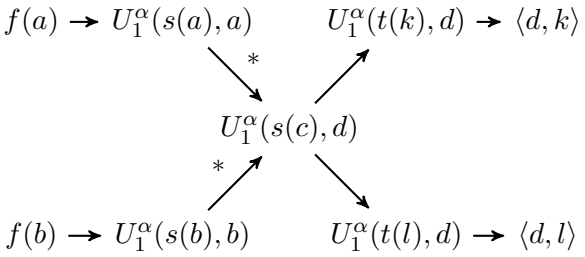

Figure 4.5: Rewrite steps in $\mathbb{U}(\mathcal{R})$ in Example 22

In [7] we furthermore showed soundness of $\mathbb{U}_{\mathsf{sim}}$ for confluent normal 1-CTRSs. Again, this result does not hold for DCTRSs:

Example 23 (Unsoundness for confluent DCTRSs)**.** *Consider the following DCTRS that is an extension of the DCTRS in 22. Like in the previous example* $\mathbb{U}_{\mathsf{seq}}$ *and* $\mathbb{U}_{\mathsf{opt}}$ *return the same TRS.*

$$\mathcal{R} = \left\{ \begin{array}{r} a \to c \searrow \\ \quad \searrow\!\nearrow e \\ b \to d \nearrow \\ k \searrow \\ \quad e \\ l \nearrow \\ s(c) \nearrow t(k) \\ \quad \searrow t(l) \\ s(e) \to t(e) \\ g(x,x) \to h(x,x) \\ f(x) \to \langle x, y\rangle \Leftarrow s(x) \to^* t(y) \end{array} \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{r} a \to c \searrow \\ \quad \searrow\!\nearrow e \\ b \to d \nearrow \\ k \searrow \\ \quad e \\ l \nearrow \\ s(c) \nearrow t(k) \\ \quad \searrow t(l) \\ s(e) \to t(e) \\ g(x,x) \to h(x,x) \\ f(x) \to U_1^\alpha(s(x), x) \\ U_1^\alpha(t(y), x) \to \langle x, y\rangle \end{array} \right\}$$

The CTRS is confluent but the derivation $g(f(a), f(b)) \to^* h(\langle d, k\rangle, \langle d, l\rangle)$ is still unsound by the same argument as in Example 22. Yet, this does not contradict soundness w.r.t. joinability:

$$g(f(a), f(b)) \to^* g(f(e), f(e)) \to^* h(f(e), f(e)) \to^* h(\langle e, e\rangle, \langle e, e\rangle)$$
$$h(\langle d, k\rangle, \langle d, l\rangle) \to^* h(\langle e, e\rangle, \langle e, e\rangle)$$

4.5 Extraction of Rewrite Sequences

In order to prove soundness of derivations we have to translate rewrite steps in arguments of U-terms into evaluations of conditions in conditional rewrite steps. For this purpose, we must extract rewrite steps from the conditional argument and the variable arguments of U-terms and additionally show that rewrite steps inside variable arguments do not change the status of the satisfiability of the conditions.

For this purpose we need to find all introduction and elimination steps of a U-term in a derivation. Therefore we define two mappings ti (for "trace introduction") and te (for "trace elimination").

For a given rewrite sequence $A : u_1 \to^* u_n$, the mapping ti returns in which rewrite step of A and at which position an introduction step is applied to an ancestor of a U-term, i.e., if $u_m|_p$ is the U-term $U^\alpha_j(v, \overrightarrow{X_j}\sigma)$, then $\mathsf{ti}(A, m, p)$ returns an index i and position q such that $u_i|_q \to^* u_m|_p$ and the $i-1$st rewrite step in A is an introduction step of which $u_i|_q$ is the contracted redex. Therefore, $u_i|_q = U^\alpha_1(s_1\tau, \overrightarrow{X_1}\tau)$ and $u_{i-1}|_q = l\tau$ where l is the left-hand side of α. Observe that there may be multiple introduction steps because of non-linear rules.

We also introduce an auxiliary mapping ti_a that traces U-terms to the closest preceding switch or introduction step(s), i.e., if $(i, q) \in \mathsf{ti}_\mathsf{a}(A, m, p)$ then $u_i|_q = U^\alpha_j(s_j\theta, \overrightarrow{X_j}\theta)$.

Definition 12 (ti)**.** *The mapping $\mathsf{ti}(A, m, p)$ of a derivation $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$, an index $m \in \{1, \ldots, n+1\}$ and a position p maps the U-term at the given position to all preceding introduction steps:*

$$\mathsf{ti}(A, m, p) = \begin{cases} \textit{undefined} & \textit{if } u_m|_p \textit{ is not a U-term} \\ \{(m, p)\} & \textit{if } p_m = p \textit{ and } \mathsf{root}(u_{m-1}|_p) \textit{ is not a U-symbol} \\ \mathsf{ti}(A, m-1, p) & \textit{if } p_m = p \textit{ and } \mathsf{root}(u_{m-1}|_p) \textit{ is a U-symbol} \\ \bigcup_{q \in Q} \mathsf{ti}(A, m-1, q) & \textit{otherwise where } Q \textit{ are all one-step ancestors of } p \end{cases}$$

The auxiliary mapping ti_a maps the U-term to all immediately preceding switch or introduction steps:

$$\mathsf{ti}_a(A, m, p) = \begin{cases} \textit{undefined} & \textit{if } u_m|_p \textit{ is not a U-term} \\ \{(m, p)\} & \textit{if } p_m = p \\ \bigcup_{q \in Q} \mathsf{ti}_a(A, m-1, q) & \textit{otherwise where } Q \textit{ are all one-step ancestors of } p \end{cases}$$

The mapping te is the symmetric counterpart of ti: It returns the index and position of the succeeding elimination step for a U-term in a derivation. Therefore, if $u_m|_p$ is the U-term

$U^\alpha_j(v, \overrightarrow{X_j}\sigma)$ and $(i,q) \in \mathsf{te}(A,m,p)$, then $u_i|_q = U^\alpha_k(t_k\tau, \overrightarrow{X_k}\tau)$ and $u_{i+1}|_q = r\tau$. Observe that there may be multiple elimination steps or none at all because of non-linear and erasing rules. We also introduce the auxiliary mapping te_a that returns the closest succeeding switch or elimination step.

Definition 13 (te)**.** *The mapping $\mathsf{te}(A,m,p)$ of a derivation $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$, an index $m \in \{1,\dots,n+1\}$ and a position p maps the U-term at the given position to all succeeding elimination steps:*

$$\mathsf{te}(A,m,p) = \begin{cases} \textit{undefined} & \textit{if } u_m|_p \textit{ is not a U-term} \\ \{(m,p)\} & \textit{if } p_m = p \textit{ and } \mathsf{root}(u_{m+1}|_p) \textit{ is not a U-symbol} \\ \mathsf{te}(A,m+1,p) & \textit{if } p_m = p \textit{ and } \mathsf{root}(u_{m+1}|_p) \textit{ is a U-symbol} \\ \bigcup_{q\in Q} \mathsf{te}(A,m+1,q) & \textit{otherwise where } Q \textit{ are all one-step descendants of } p \end{cases}$$

The auxiliary mapping te_a maps the U-term to all immediately succeeding switch or elimination steps:

$$\mathsf{te}_a(A,m,p) = \begin{cases} \textit{undefined} & \textit{if } u_m|_p \textit{ is not a U-term} \\ \{(m,p)\} & \textit{if } p_{m+1} = p \\ \bigcup_{q\in Q} \mathsf{te}_a(A,m+1,q) & \textit{otherwise where } Q \textit{ are all one-step descendants of } p \end{cases}$$

In derivations in DCTRSs that start from an original term, all U-terms must be preceded by at least one introduction step. Therefore $|\mathsf{ti}(A,m,p)| \geq 1$ for all U-terms $u_m|_p$ in such a derivation. If $|\mathsf{ti}(A,m,p)| = 1$, then $u_m|_p$ is *uniquely introduced*.

If $\mathsf{te}(A,m,p) = \emptyset$ for a U-term $u_m|_p$, then the U-term is *erased*, if $|\mathsf{te}(A,m,p)| \geq 1$ it is *eliminated*, and if $|\mathsf{te}(A,m,p)| = 1$ it is *uniquely eliminated*. Since U-terms in the last term of a derivation do not have a one-step descendant we consider them being erased.

We call $u_{i-1}|_q = l\sigma$ an *introduction term* for all $(i,q) \in \mathsf{ti}(A,m,p)$. Dually, for all $(i,q) \in \mathsf{te}(A,m,p)$, $u_{i+1}|_q = r\tau$ is an *elimination term*. The redexes of switch rules are *switch terms*.

Example 24. *The unraveled TRS of Example 12 is*

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{r} x + 0 \to x \\ x + s(y) \to s(x+y) \\ quad(x) \to U^\alpha_1(x+x,x) \\ U^\alpha_1(y,x) \to U^\alpha_2(y+y,x,y) \\ U^\alpha_2(z,x,y) \to z \end{array} \right\}$$

In the following derivation we can trace all U-terms. For better readability we label the positions of U-terms in the following derivation:

$$\begin{aligned} quad(s(0)) &\to \overbrace{U^\alpha_1(s(0)+s(0),s(0))}^{(1)} \to^* \overbrace{U^\alpha_1(s^2(0),s(0))}^{(2)} \\ &\to \underbrace{U^\alpha_2(s^2(0)+s^2(0),s(0),s^2(0))}_{(3)} \to \underbrace{U^\alpha_2(s^4(0),s(0),s^2(0))}_{(4)} \to s^4(0) \end{aligned}$$

ti_a *of the U-term at* (2) *returns* (1) *because it is the first introduced U-term.* ti_a *of* (4) *returns* (3)*, the first U-term after the switch step.* te_a *of* (1) *is* (2)*, and* te_a *of* (3) *is* (4)*.*

ti *of* (4) *returns the U-term at* (1)*, therefore* $quad(s(0))$ *is the introduction term of* (4)*, and* (4) *is uniquely introduced.*

te *for* (1) *returns* (4) *so that* $s^4(0)$ *is the elimination term.* (1) *therefore is uniquely eliminated.*

For the unsound derivation of Example 20 tracing U-terms is complex because some U-terms have multiple one-step descendants and one-step ancestors, and some are erased:

Example 25. *The unsound derivation of Example 20 contains U-terms that have multiple one-step ancestors and one-step descendants:*

$$\begin{aligned} g(f(a),f(b)) &\to g(\overbrace{U_1^\alpha(a,a)}^{(1)},f(b)) \to g(U_1^\alpha(a,a),\overbrace{U_1^\alpha(b,b)}^{(2)}) \to^* g(\overbrace{U_1^\alpha(c,d)}^{(3)},\overbrace{U_1^\alpha(c,d)}^{(4)}) \\ &\to h(\underbrace{U_1^\alpha(c,d)}_{(5)},\underbrace{U_1^\alpha(c,d)}_{(6)},f(k)) \to h(d,U_1^\alpha(c,d),f(k)) \to^* h(d,\underbrace{U_1^\alpha(k,k)}_{(7)},U_1^\alpha(k,k)) \to A \end{aligned}$$

ti *of the U-term at* (3) *is* $\{(1)\}$*, and* ti *of* (4) *is* $\{(2)\}$*.* ti *of* (5)*,* (6) *and* (7) *returns* $\{(1),(2)\}$*. Therefore, these U-terms are not uniquely introduced.*

The U-terms at (6) *and* (7) *are erased while the U-term at* (5) *(and therefore also* (1)*,* (2)*,* (3) *and* (4)*) is eliminated.* te *of* (6) *and* (7) *therefore is the empty set and* te *of* (5) *is* (5) *itself.*

After we traced derivations of U-terms we can extract subderivations from the conditional argument and the variable arguments:

Lemma 2. *Let* $\mathbb{U}(\mathcal{R})$ *be the unraveling of some DCTRS* $\mathcal{R}$ *and* $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ *be a derivation in* $\mathbb{U}(\mathcal{R})$*. Let* $u_m|_p = U_j^\alpha(v,\overrightarrow{X_j}\tau)$ *and* $(i,q) \in \mathsf{ti}_a(A,m,p)$*. Then* $u_i|_q = U_j^\alpha(s_j\sigma_j,\overrightarrow{X_j}\sigma_j)$ *and* $u_i|_q \to^*_{\mathbb{U}(\mathcal{R})} u_m|_p$*.*

Dually, if $(i,q) \in \mathsf{te}_a(A,m,p)$*, then* $u_i|_q = U_j^\alpha(t_j\sigma_{j+1},\overrightarrow{X_j}\sigma_{j+1})$ *and* $u_m|_p \to^*_{\mathbb{U}(\mathcal{R})} u_i|_q$*.*

Proof. This result is an immediate consequence of Definition 12 and Definition 13, and the definition of one-step ancestors and one-step descendants. □

The following lemma shows how we can extract derivations in conditional arguments and variable arguments in U-terms using ti and te.

Lemma 3. *Let* $\mathbb{U}(\mathcal{R})$ *be the unraveling of some DCTRS* $\mathcal{R}$ *and* $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ *be a derivation in* $\mathbb{U}(\mathcal{R})$*. Let* $u_m|_p = U_j^\alpha(v,\overrightarrow{X_j}\tau)$*,* $(i,q) \in \mathsf{ti}_a(A,m,p)$ *and* $u_i|_q = U_j^\alpha(s_j\sigma_j,\overrightarrow{X_j}\sigma_j)$*. Then* $s_j\sigma_j \to^*_{\mathbb{U}(\mathcal{R})} v$ *and* $x\sigma_j \to^*_{\mathbb{U}(\mathcal{R})} x\tau$ *for all* $x \in X_j$*.*

Dually, if $(i,q) \in \mathsf{te}_a(A,m,p)$ *and* $u_i|_q = U_j^\alpha(t_j\sigma_{j+1},\overrightarrow{X_j}\sigma_{j+1})$*, then* $v \to^*_{\mathbb{U}(\mathcal{R})} t_j\sigma_{j+1}$ *and* $x\tau \to^*_{\mathbb{U}(\mathcal{R})} x\sigma_{j+1}$ *for all* $x \in X_j$*.*

Proof. Let $(i,q) \in \mathsf{ti}_\mathsf{a}(A,m,p)$. We proof this lemma by induction on $m-i$ for ti_a.

If $m-i=0$, then $\mathsf{ti}_\mathsf{a}(A,m,p)=\{(m,p)\}$, $U_j^\alpha(v,\overrightarrow{X_j}\tau)=U_j^\alpha(s_j\sigma_j,\overrightarrow{X_j}\sigma_j)$ and therefore by Lemma 2 $v=s_j\sigma_j$ and $x\tau=x\sigma_j$ for all $x\in X_j$.

Otherwise, $m-i\geq 1$. In this case $p_{i-1}\neq p$ and $\mathsf{ti}_\mathsf{a}(A,m,p)=\bigcup_{p'\in P}\mathsf{ti}_\mathsf{a}(A,m-1,p')$ where P are all one-step ancestors of p.

Let $p'\in P$ be a one-step ancestor of $u_m|_p$ such that $(i,q)\in\mathsf{ti}_\mathsf{a}(A,m-1,p')$ and $u_{m-1}|_{p'}=U_j^\alpha(v',\overrightarrow{X_j}\tau')$. By the inductive hypothesis we obtain $s_j\sigma_j\to^*_{\mathbb{U}(\mathcal{R})}v'$ and $x\sigma_j\to^*_{\mathbb{U}(\mathcal{R})}x\tau'$.

By the definition of one-step ancestors we obtain $U_j^\alpha(v',\overrightarrow{X_j}\tau')\to^{\leq 1}_{\mathbb{U}(\mathcal{R})}U_j^\alpha(v,\overrightarrow{X_j}\tau)$, therefore $v'\to^{\leq 1}_{\mathbb{U}(\mathcal{R})}v$ and $x\tau'\to^{\leq 1}_{\mathbb{U}(\mathcal{R})}x\tau$ for all $x\in X_j$.

For te_a, the proof is analogously. Let $(i,q)\in\mathsf{te}_\mathsf{a}(A,m,p)$. By induction on $i-m$: If $i-m=0$, then $\mathsf{te}_\mathsf{a}(A,m,p)=\{(m,p)\}$.

Otherwise, let p' be a one-step descendants p such that $(i,q)\in\mathsf{te}_\mathsf{a}(A,m+1,p')$. By the inductive hypothesis, $v'\to^* t_j\sigma_{j+1}$ and $x\tau'\to^* x\sigma_{j+1}$ where $u_{m+1}|_{p'}=U_j^\alpha(v',\overrightarrow{X_j}\tau')$ and $u_i|_q=U_j^\alpha(t_j\sigma_{j+1},\overrightarrow{X_j}\sigma_{j+1})$.

Since also $v\to^{\leq 1}_{\mathbb{U}(\mathcal{R})}v'$ and $x\tau\to^{\leq 1}_{\mathbb{U}(\mathcal{R})}x\tau'$ for all $x\in X_j$ our result holds. □

We can immediately extend this result to entire rewrite sequences from introduction terms to elimination terms:

Lemma 4 (extraction of arguments in U-terms)**.** *Let $\mathbb{U}(\mathcal{R})$ be the unraveling of some DCTRS $\mathcal{R}$ and $A: u_1\to_{p_1}u_2\cdots\to_{p_n}u_{n+1}$ be a derivation in $\mathbb{U}(\mathcal{R})$. Let $u_m|_p$ be a U-term of the conditional rule $\alpha: l\to r\Leftarrow s_1\to^* t_1,\ldots,s_k\to^* t_k$, let $(i,q)\in\mathsf{ti}(A,m,p)$ such that $u_{i-1}|_q=l\sigma_1$ is an introduction term, and let $(j,o)\in\mathsf{te}(A,m,p)$ such that $u_{j+1}|_o=r\sigma_{k+1}$ is an elimination term of $u_m|_p$.*

Then there are matchers $\sigma_2,\ldots,\sigma_k$ such that $x\sigma_1\to^* x\sigma_2$, …, $x\sigma_k\to^* x\sigma_{k+1}$ (for all $x\in\mathcal{V}ar(\alpha)$), and $s_1\sigma_1\to^* t_1\sigma_2$, …, $s_k\sigma_k\to^* t_k\sigma_{k+1}$ in $\mathbb{U}(\mathcal{R})$.

Proof. By repeated application of Lemma 2 and Lemma 3, we obtain a derivation

$$\begin{aligned}
u_{i-1}|_q=l\sigma_1&\to U_1^\alpha(s_1\sigma_1,\overrightarrow{X_1}\sigma_1))\\
&\to^* U_1^\alpha(t_1\sigma_2,\overrightarrow{X_1}\sigma_2)\to U_2^\alpha(s_2\sigma_2,\overrightarrow{X_2}\sigma_2)\\
&\to^* U_2^\alpha(t_2\sigma_3,\overrightarrow{X_2}\sigma_3)\to U_3^\alpha(s_3\sigma_4,\overrightarrow{X_3}\sigma_3)\\
&\quad\vdots\qquad\vdots\\
&\to^* U_k^\alpha(t_k\sigma_{k+1},\vec{X}_k\sigma_{k+1})\to r\sigma_{k+1}=u_{j+1}|_o
\end{aligned}$$

and hence the derivations $s_i\sigma_i\to^*_{\mathbb{U}_\mathsf{opt}(\mathcal{R})}t_i\sigma_{i+1}$ and $x_i\sigma_i\to^*_{\mathbb{U}_\mathsf{opt}(\mathcal{R})}x\sigma_{i+1}$ for all $x\in X_i$ ($i\in\{2,\ldots,k\}$). □

Observe that the sum of the length of all extracted derivations is smaller than the number of rewrite steps between the introduction term and the U-term.

The previous lemmata prove that we can extract derivations from the conditional and the variable arguments of U-terms. This will play an important role in our soundness proofs.

4.6 Term Translations

One approach is to prove soundness using the backtranslation tb and the translation forward tf. For this purpose we need to prove that we can translate all rewrite steps in a derivation in a transformed CTRS into valid conditional rewrite sequences. This is not possible in general. Example 21 and Example 6 are counterexamples for tb. Also for tf this is not possible:

Example 26. *Consider the following CTRS and its unraveling:*

$$\mathcal{R} = \left\{ \begin{array}{c} a \\ b \end{array} \begin{array}{c} \searrow \\ \nearrow \end{array} c \quad f(x) \to x \Leftarrow a \to^* x \right\} \qquad \mathbb{U}(\mathcal{R}) = \left\{ \begin{array}{c} a \\ b \end{array} \begin{array}{c} \searrow \\ \nearrow \end{array} c \quad f(x) \to U_1^\alpha(a,x) \quad U_1^\alpha(x,x) \to x \right\}$$

Consider the derivation

$$f(b) \to U_1^\alpha(a,b) \to U_1^\alpha(c,b) \to U_1^\alpha(c,c) \to c.$$

This derivation is sound because $f(b) \to_\mathcal{R} f(c) \to_\mathcal{R} c$. The first rewrite step of the derivation $f(b) \to U_1^\alpha(a,b)$ is translated into $f(b) \to b$ using tf*, yet this is not a valid rewrite step in $\mathcal{R}$ because the condition $a \to^* b$ is not satisfied.*

Nevertheless we will analyze in the following in which cases we obtain valid derivations in the original CTRS if we translate rewrite steps in derivations using tf and tb.

4.6.1 Right-Separate CTRSs

Consider a derivation $u_1 \to u_2 \to \cdots u_n$ in some transformed TRS $\mathbb{U}(\mathcal{R})$. In general translating all single rewrite steps $\mathsf{tf}(u_i) \to^* \mathsf{tf}(u_{i+1})$ does not yield valid derivations in the original CTRS $\mathcal{R}$, in particular not if u_i contains a U-term for which the condition is not satisfied.

Example 27. *Consider the CTRS of Example 5 and its unraveled TRS using $\mathbb{U}_{\mathsf{opt}}$:*

$$\mathbb{U}_{\mathsf{opt}}(or(x,y) \to true \Leftarrow x \to^* true) = \left\{ \begin{array}{l} or(x,y) \to U_1^\alpha(x) \\ U_1^\alpha(true) \to true \end{array} \right\}$$

We obtain the following rewrite step in the transformed TRS:

$$or(false, false) \to U^\alpha(false)$$

We cannot translate this rewrite step into a valid conditional rewrite step with tf *because* $\mathsf{tf}(U^\alpha(false)) = true$.

Translating U-terms forward is only useful if the condition is eventually satisfied, i.e., if the U-term to which tf is applied if eliminated. This is satisfied if $\mathcal{R}$ is ultra-non-erasing, and in fact we have shown in [7] that $\mathbb{U}_{\mathsf{sim}}$ is sound for (ultra-)non-erasing normal 1-CTRSs. Yet for DCTRSs we obtain unsoundness for ultra-non-erasingness and sortedness (see Example 22).

tf is not properly definable for 3-CTRSs because the right-hand side of conditional rules may contain extra variables that are not encoded in all U-terms (see Example 17). tf is therefore only properly defined for 2-CTRSs. Yet, we obtain unsoundness even for sorted, ultra-non-erasing 2-DCTRSs:

Example 28. *Consider the following sorted, ultra-non-erasing, deterministic 2-CTRS and its unraveled TRS using* $\mathbb{U}_{\mathsf{opt}}$*:*

$$\mathcal{R} = \left\{ \begin{array}{c} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ s(k) \searrow t(a) \\ s(l) \nearrow \\ g(x,x) \to h(x,x) \\ f(x,y) \to y \Leftarrow s(x) \to^* t(y) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{c} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ s(k) \searrow t(a) \\ s(l) \nearrow \\ g(x,x) \to h(x,x) \\ f(x,y) \to U_1^\alpha(s(x),y) \\ U_1^\alpha(t(y),y) \to y \end{array} \right\}$$

Consider the following derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$*:*

$$\begin{aligned} g(f(k,b),f(l,b)) &\Vdash g(U_1^\alpha(s(k),b),U_1^\alpha(s(l),b)) \Vdash g(U_1^\alpha(t(a),b),U_1^\alpha(t(a),b)) \\ &\to h(U_1^\alpha(t(a),b),U_1^\alpha(t(a),b)) \Vdash h(U_1^\alpha(t(c),c),U_1^\alpha(t(d),d)) \Vdash h(c,d) \end{aligned}$$

This derivation is unsound for the following reason: Consider some f*-rooted term* $f(u,v)$*. The condition is only satisfiable if* $u \in \{k,l\}$*, and* v *is* a *or some successor of* a*.*

Consider the term $g(f(k,b),f(l,b))$*. Before applying the non-linear rule we need to apply the conditional rule to both* f*-rooted terms since* k *and* l *do not have a common reduct.*

The condition in both f*-terms is not satisfiable because* $a \not\to^* b$*. Yet, we can rewrite* b *to* c *and* d*. Both of these terms are also successors of* a*, hence the conditions are satisfiable.*

In both f*-terms,* b *must rewrite to the same term because otherwise we cannot apply the non-linear rule. Therefore the only possible rewrite sequences in* $\mathcal{R}$ *are*

$$\begin{aligned} g(f(k,b),f(l,b)) &\to^* g(f(k,c),f(l,c)) \to^* g(c,c) \to h(c,c) \\ g(f(k,b),f(l,b)) &\to^* g(f(k,d),f(l,d)) \to^* g(d,d) \to h(d,d) \\ g(f(k,b),f(l,b)) &\to^* g(f(k,c),f(l,d)) \to^* g(c,d) \not\to h(c,d) \end{aligned}$$

Therefore, the derivation is unsound.

In the previous example the term $g(U_1^\alpha(t(a),b),U_1^\alpha(t(a),b))$ contains two equal U-terms. Yet, their introduction terms are different: $f(k,b)$ and $f(l,b)$. This is not a problem if they give rise to the same elimination term, yet the condition only becomes satisfiable after the variable argument has been rewritten to c or d. In the elimination step, it therefore seems that the variable argument in the introduction term already contained c or d while it was b.

We need to avoid this asynchronicity between the conditional argument and the variable arguments. For this purpose we define a variable condition that is stronger than sortedness: If a

condition in a conditional rule α is not satisfied for some σ, then it must remain unsatisfied for all τ where $x\sigma \to^* x\tau$ for all $x \in \mathcal{V}ar(\alpha)$.

This is only satisfied if the left-hand side of a conditional rule does not share variables with the right-hand sides of the conditions since otherwise one rewrite step in the variable argument can flip the status of satisfiability. If variables in the right-hand sides of conditions do not occur in variable arguments this is not possible.

In order to also preserve the bindings of variables on the right-hand side of the conditional rule the right-hand sides of the conditions also must not share variables with the right-hand side of the rule. We call this property *right-separateness* because the variables on the right-hand sides of the conditions must be separate from all other variables:[1]

Definition 14 (right-separateness)**.** *A CTRS $\mathcal{R}$ is right-separate if for all conditional rules α : $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k \in \mathcal{R}$, $t_i \cap \mathcal{V}ar(l) = \emptyset$, $\mathcal{V}ar(t_i) \cap \mathcal{V}ar(r) = \emptyset$ and $\mathcal{V}ar(t_i) \cap \mathcal{V}ar(t_j) = \emptyset$ if $i \neq j$ for all $i, j \in \{1, \ldots, k\}$.*

Using right-separateness, we know that if a U-term is eliminated, then its conditions are already satisfied for the variable bindings used in the introduction step. Furthermore, right-separateness of a deterministic CTRS implies that all variables on the right-hand side of a conditional rule already occur on the left-hand side and that the CTRS is sorted.

Because of ultra-non-erasingness, all U-terms are eventually eliminated. We therefore can replace all U-terms by the corresponding original term using the translation-forward $\mathtt{tf}$.

The CTRS in Example 28 is not right-separate because for the conditional rule $\mathcal{V}ar(t(y)) \cap \mathcal{V}arr \neq \emptyset$:

$$f(x, y) \to y \Leftarrow s(x) \to^* t(y)$$

Applying $\mathtt{tf}$ to all U-terms in the unsound derivation in this example returns the derivation:

$$g(f(k, b), f(l, b)) \to^* g(b, b) \to^* h(b, b) \to^* h(c, d)$$

Yet, $f(k, b)$ does not rewrite to b in the original CTRS.

In order to prove soundness of $\mathbb{U}_{\mathsf{opt}}$ for right-separate, ultra-non-erasing DCTRSs we follow the proof structure of [8]:

First we prove a monotony property of $\mathtt{tf}$.

Lemma 5 (monotony property of $\mathtt{tf}$ [8, Lemma 15])**.** *Let $u \to_p v$ be a rewrite step in a derivation of some unraveled CTRS $\mathbb{U}(\mathcal{R})$. Let furthermore $\mathtt{tf}(u|_p) \to^*_{\mathcal{R}} \mathtt{tf}(v|_p)$. Then $\mathtt{tf}(u|_q) \to^*_{\mathcal{R}} \mathtt{tf}(v|_{q'})$ for all one-step-descendants $v|_{q'}$ of $u|_q$.*

Proof. By case distinction on p and q: The case $p = q$ is trivial.

If $p \parallel q$, then the only one-step descendant of q is q itself and $u|_q = v|_q$, hence $\mathtt{tf}(u|_q) = \mathtt{tf}(v|_q)$.

If $p < q$, then for all one-step descendants q' of q $u|_q = v|_{q'}$ and therefore also $\mathtt{tf}(u|_q) = \mathtt{tf}(v|_q)$.

Finally, if $q < p$, then $q = q'$. We use induction on $|p| - |q|$. If $|p| - |q| = 0$ our result holds vacuously.

[1] In [8] we called this property almost-right-stability.

For the inductive step let $p = q.i.o$ where $i \in \mathbb{N}$. We distinguish two cases:
If $u|_q = f(u_1, \ldots, u_n)$ where $f \in \mathcal{F}$ is an original symbol then

$$v|_q = f(u_1, \ldots, u_{i-1}, v_i, u_{i+1}, \ldots, u_n).$$

Furthermore, $u_i \to_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v_i$. By the inductive hypothesis, $\mathsf{tf}(u_i) \to^*_{\mathcal{R}} \mathsf{tf}(v_i)$, which implies

$$\mathsf{tf}(f(u_1, \ldots, u_n)) \to^*_{\mathcal{R}} \mathsf{tf}(f(u_1, \ldots, u_{i-1}, v_i, u_{i+1}, \ldots, u_n))$$

If otherwise $u|_q = U^\alpha_j(u', \overrightarrow{X_j}\sigma)$ is a U-term, then $v|_q = U^\alpha_j(v', \overrightarrow{X_j}\tau)$ where α is the rule $l \to r \Leftarrow c$. In this case $\mathsf{tf}(u|_q) = r\,\mathsf{tf}(\sigma)$ and $\mathsf{tf}(v|_q) = r\,\mathsf{tf}(\tau)$. We therefore need to show that $r\,\mathsf{tf}(\sigma) \to^*_{\mathcal{R}} r\,\mathsf{tf}(\tau)$.

We distinguish two cases whether i is a conditional or a variable position. If $i = 1$, then the rewrite step is inside the conditional argument and therefore $\sigma = \tau$. Therefore also $\mathsf{tf}(\sigma) = \mathsf{tf}(\tau)$ and $\mathsf{tf}(u|_q) = \mathsf{tf}(v|_q)$.

Otherwise, the rewrite step is inside a variable argument so that $x\sigma \to_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} x\tau$ for $x = \overrightarrow{X_j}[i-1]$ and $y\sigma = y\tau$ for all $y \in X_j \setminus \{x\}$. By the inductive hypothesis, $x\,\mathsf{tf}(\sigma) \to^*_{\mathcal{R}} x\,\mathsf{tf}(\tau)$. Therefore, for all $P = \{p' \in \mathcal{VP}os(r) \mid r|_{p'} = x\}$

$$r\,\mathsf{tf}(\sigma) \Vdash_{\mathcal{R}} r\,\mathsf{tf}(\sigma)[x\,\mathsf{tf}(\tau)]_P = r\,\mathsf{tf}(\tau)$$

□

Next we prove that we can translate single rewrite steps in derivations:
correct number

Lemma 6 (technical key result [8, Lemma 16])**.** *Let $\mathcal{R}$ be an $\mathbb{U}_{\mathsf{opt}}$-non-erasing and right-separate DCTRS and let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ be a derivation such that $u_{n+1} \in \mathcal{T}$. Then $\mathsf{tf}(u_m|_{p_m}) \to^*_{\mathcal{R}} \mathsf{tf}(u_{m+1}|_{p_m})$ for all $m \in \{1, \ldots, n\}$.*

Proof. We prove this result by induction on the length n of A.

If A has length 1, then we distinguish two cases:

If the applied rule is an original rule $l \to r$, then $u_n|_{p_n} = l\sigma$ and $u_{n+1}|_{p_n} = r\sigma$. Since $l \to r \in \mathcal{R}$ also $l\,\mathsf{tf}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tf}(\sigma)$.

If the applied rule is an elimination rule, then $u_{n+1}|_{p_n} = r\sigma$ and $u_n|_{p_n} = U^\alpha_k(t_k\sigma, \overrightarrow{X_k}\sigma)$. By the definition of tf, $\mathsf{tf}(u_n|_{p_n}) = r\,\mathsf{tf}(\sigma) = \mathsf{tf}(u_{n+1}|_{p_n})$.

For the inductive step observe that by the inductive hypothesis $\mathsf{tf}(u_m|_{p_m}) \to^*_{\mathcal{R}} \mathsf{tf}(u_{m+1}|_{p_m})$ for all $m \in \{2, \ldots, n\}$. Therefore, Lemma 5 holds for these rewrite steps.

We use a case distinction on the first rewrite step of A:

If the applied rule is an original rule $l \to r$, then $u_1|_{p_1} = l\sigma$ and $u_2|_{p_1} = r\sigma$. Since $l \to r \in \mathcal{R}$ also $l\,\mathsf{tf}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tf}(\sigma)$.

If the applied rule is a switch- or elimination rule, then $\mathsf{tf}(u_1|_{p_1}) = \mathsf{tf}(u_2|_{p_1})$. Observe that in the case of switch rules all variable bindings of r are preserved.

Finally, if it is an introduction rule, then $u_1|_{p_1} = l\sigma_1$, $u_2|_{p_1} = U^\alpha_1(s_1\sigma_1, \overrightarrow{X_1}\sigma_1)$, $\mathsf{tf}(u_1|_{p_1}) = l\,\mathsf{tf}(\sigma_1)$ and $\mathsf{tf}(u_2|_{p_1}) = r\,\mathsf{tf}(\sigma_1)$ where α is the rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$.

Let $(m, q) \in \mathsf{te}(A, 2, p_1)$. By Lemma 4 the derivation A contains the derivations $s_i\sigma_i \to^* t_i\sigma_{i+1}$ and $x\sigma_i \to^* x\sigma_{i+1}$ ($x \in X_i$) for all $i \in \{1, \ldots, k\}$ from A.

By repeated application of the inductive hypothesis and Lemma 5, $s_i\,\mathsf{tf}(\sigma_i) \to^*_{\mathcal{R}} t_i\,\mathsf{tf}(\sigma_{i+1})$ and $x\,\mathsf{tf}(\sigma_i) \to^*_{\mathcal{R}} x\,\mathsf{tf}(\sigma_{i+1})$.

Let in the following $\sigma = \sigma_1 \circ \sigma_2 \circ \cdots \circ \sigma_{k+1}$ be a combination of all substitutions from left to right. Observe that $l\sigma_1 = l\sigma$, $s_i\sigma \to^* s_i\sigma_i$ and by right-separateness and the definition of deterministic extra variables $t_i\sigma_{i+1} = t_i\sigma$.

We therefore obtain the following derivations in $\mathcal{R}$:

$$s_i\,\mathsf{tf}(\sigma) \to^*_{\mathcal{R}} s_i\,\mathsf{tf}(\sigma_i) \to^*_{\mathcal{R}} t_i\,\mathsf{tf}(\sigma_i)) \to^*_{\mathcal{R}} t_i\,\mathsf{tf}(\sigma) \text{ for all } i \in \{1, \ldots, k\}$$

Therefore, the conditions are satisfied for $\mathsf{tf}(\sigma)$ so that $l\,\mathsf{tf}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tf}(\sigma)$. Therefore $\mathsf{tf}(u_1|_{p_1}) \to_{\mathcal{R}} \mathsf{tf}(u_2|_{p_1})$. □

Using the two previous technical lemmata we can prove soundness:
correct number TODO p54 put in comments

Lemma 7 (soundness for ultra-non-erasing, right-separate DCTRSs [8, Lemma 17])**.** *Let $\mathcal{R}$ be a $\mathbb{U}_{\mathsf{opt}}$-non-erasing, right-separate 2-DCTRS, and $A : u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$, $v \in \mathcal{T}$ be a derivation in its unraveled TRS, then $\mathsf{tf}(u) \to^*_{\mathcal{R}} v$.*

Proof. Let A be the derivation $u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$. By Lemma 6, $\mathsf{tf}(u_i|_{p_i}) \to^*_{\mathcal{R}} \mathsf{tf}(u_{i+1}|_{p_i})$ for all $i \in \{1, \ldots, n\}$. By Lemma 5, $\mathsf{tf}(u_i|_\epsilon) \to^*_{\mathcal{R}} \mathsf{tf}(u_{i+1}|_\epsilon)$. Since $\mathsf{tf}(u_{n+1}) = u_{n+1}$, this implies $\mathsf{tf}(u_1) \to^*_{\mathcal{R}} u_{n+1}$. □

Theorem 1 (soundness for ultra-non-erasing, right-separate DCTRSs [8, Theorem 18])**.** *Let $\mathcal{R}$ be an $\mathbb{U}_{\mathsf{opt}}$-non-erasing and right-separate 2-DCTRS, then $\mathbb{U}_{\mathsf{opt}}$ is sound.*

Proof. Straightforward from Lemma 7. □

Observe that this result does not imply soundness w.r.t. joinability (see Example 21).

4.6.2 Left-Separated CTRSs

In right-separate CTRSs the conditions are satisfied for the variable bindings in the introduction term. We therefore can replace all succeeding U-terms by the corresponding right-hand side of the conditional rule using tf.

If the conditions are satisfied for the variable bindings in the elimination term (but not necessarily for the introduction term), then we can replace all U-terms by the corresponding left-hand side of the conditional rule using tb. In order to ensure that this is possible, conditions must remain satisfied even after rewrite steps in the variable arguments.

Consider the unsound derivation of Example 22:

$$\begin{aligned} g(f(a), f(b)) &\to g(U_1^\alpha(s(a), a), U_1^\alpha(s(a), a) \\ &\to^* h(U_1^\alpha(t(k), d), U_1^\alpha(t(l), d) \to h(\langle d, k\rangle, \langle d, l\rangle) \end{aligned}$$

We can extract the derivation $f(a) \to U_1^\alpha(s(a), a) \to^* U_1^\alpha(t(k), d) \to \langle d, k\rangle$ from the unsound derivation. The conditional rule is $f(x) \to \langle x, y\rangle \Leftarrow s(x) \to^* t(y)$. The condition $s(x) \to^* t(y)$ is satisfied for the variable binding of the introduction term, $s(a) \to^* t(k)$, but not for the elimination term $s(d) \not\to^* t(k)$.

In order to avoid such cases we need to ensure that the left-hand sides of the conditions do not share variables with other left-hand sides of the conditions or the right-hand side of the conditional rule. This property is the symmetric counterpart of right-separateness, therefore we refer to it as *left-separateness*:

Definition 15 (left-separateness)**.** *A CTRS is left-separate if for all conditional rules $\alpha : l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k \in \mathcal{R}$, $t_i \cap \mathcal{V}ar(r) = \emptyset$, $\mathcal{V}ar(s_i) \cap \mathcal{V}ar(l) = \emptyset$ and $\mathcal{V}ar(s_i) \cap \mathcal{V}ar(s_j) = \emptyset$ if $i \neq j$ for all $i, j \in \{1, \ldots, k\}$.*

This case subsumes the soundness proof of ultra-right-linear DCTRSs in [8].

Furthermore, all U-terms must contain all variable bindings of the left-hand side. This is not satisfied for $\mathbb{U}_{\mathsf{opt}}$ in general, therefore we will use $\mathbb{U}_{\mathsf{seq}}$ instead. Since tb does not imply that the condition of a U-term must be satisfied we can also drop the requirement for ultra-non-erasingness.

Yet, left-separate systems have one major disadvantage: We cannot use variables of the left-hand side of a conditional rule in the left-hand side of a condition. Such systems are therefore of only limited practical use because we cannot apply any functions to variables of the left-hand side in the conditions. In fact, this class of deterministic CTRS is the extension of ultra-right-linear normal 1-CTRSs for which we have shown soundness in [7].

The proof is completely analogous to the proof of soundness for right-separate systems. We first prove monotony of tb:

Lemma 8 (monotony property of tb)**.** *Let $u \to_p v$ be a rewrite step in some derivation of an unraveled CTRS $\mathbb{U}(\mathcal{R})$. Let furthermore $\mathsf{tb}(u|_p) \to^*_{\mathcal{R}} \mathsf{tb}(v|_p)$, then $\mathsf{tb}(u|_q) \to^*_{\mathcal{R}} \mathsf{tb}(v|_{q'})$ for all one-step-descendants $v|_{q'}$ of $u|_q$.*

Proof. The proof is identical to the one of Lemma 5 except for the use of tb and the left-hand side of the rule l instead of the right-hand side. □

Next, we prove that we can translate every rewrite step using tb:

Lemma 9 (technical key result)**.** *Let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$, $u_1 \in \mathcal{T}$, be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a left-separate DCTRS. Then $\mathsf{tb}(u_m|_{p_m}) \to^*_{\mathcal{R}} \mathsf{tb}(u_{m+1}|_{p_m})$ for all $m \in \{1, \ldots, n\}$.*

Proof. The proof is very similar to the proof of Lemma 6: We prove this result by induction on the length n of A.

For the induction base $|A| = 1$, we either apply an unconditional rule so that $u_1|_{p_1}$ and $u_2|_{p_1}$ are original terms, or an introduction rule so that $\mathsf{tb}(u_2|_{p_1}) = u_1|_{p_1}$.

By the inductive hypothesis, $\mathsf{tb}(u_i|_{p_i}) \to^*_{\mathcal{R}} \mathsf{tb}(u_{i+1}|_{p_i})$ for $i \in \{1, \ldots, n-1\}$.

For the inductive step we distinguish several cases depending on the rule applied in the last rewrite step of A:

If the rule is an introduction- or switch rule, then $\mathsf{tb}(u_n|_{p_n}) = \mathsf{tb}(u_{n+1}|_{p_n})$.

If it is an original rule, $u_n|_{p_n} = l\sigma$ and $u_{n+1}|_{p_n} = r\sigma$, $l \to r \in \mathcal{R}$ and hence $l\,\mathsf{tb}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tb}(\sigma)$.

If it is an elimination rule, $u_n|_{p_n} = U_k^\alpha(t_k\sigma_{k+1}, \overrightarrow{X_k\sigma_{k+1}})$, $u_{n+1}|_{p_n} = r\sigma_{k+1}$, $\mathsf{tb}(u_n|_{p_n}) = l\,\mathsf{tb}(\sigma_{k+1})$ and $\mathsf{tb}(u_{n+1}|_{p_n}) = r\,\mathsf{tb}(\sigma_{k+1})$ where α is the rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$.

It remains to show that the conditions are satisfied for σ_{k+1}: We obtain the following subderivations of A by Lemma 4: $s_i\sigma_i \to^* t_i\sigma_{i+1}$ and $x\sigma_i \to^* x\sigma_{i+1}$ $(x \in X_i)$ for all $i \in \{1, \ldots, k\}$. We can apply our inductive hypothesis to these subderivations: $s_i\,\mathsf{tb}(\sigma_i) \to^*_{\mathcal{R}} t_i\,\mathsf{tb}(\sigma_{i+1})$ and $x\,\mathsf{tb}(\sigma_i) \to^*_{\mathcal{R}} x\,\mathsf{tb}(\sigma_{i+1})$.

Let in the following $\sigma = \sigma_{k+1} \circ \sigma_k \circ \cdots \circ \sigma_1$ be a combination of all substitutions from right to left, therefore $r\sigma = r\sigma_{k+1}$. Left-separateness implies $s_i\sigma_i = s_i\sigma$.

We obtain the following derivations in $\mathcal{R}$: $s_i\,\mathsf{tb}(\sigma) \to^*_{\mathcal{R}} t_i\,\mathsf{tb}(\sigma_i) \to^*_{\mathcal{R}} t_i\,\mathsf{tb}(\sigma)$ for all $i \in \{1, \ldots, k\}$, i.e., the conditions are satisfied for $\mathsf{tb}(\sigma)$ in $\mathcal{R}$. Therefore, $l\,\mathsf{tb}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tb}(\sigma)$. □

Using the two previous technical lemmata we can prove soundness for left-separate DCTRSs:

Lemma 10 (soundness of left-separate DCTRSs)**.** *Let $\mathcal{R}$ be a left-separate DCTRS, and let $A : u \to^*_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} v$, $u \in \mathcal{T}$ be a derivation in its unraveled TRS, then $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$.*

Proof. The proof is analogous to the proof of Lemma 7, via Lemma 9 and Lemma 8. □

Theorem 2 (soundness of left-separate DCTRSs)**.** *Let $\mathcal{R}$ be a left-separate DCTRS. Then $\mathbb{U}_{\mathsf{seq}}$ is sound.*

Proof. Straightforward from Lemma 10. □

For left-separate DCTRSs we even obtain soundness w.r.t. joinability:

Lemma 11 (soundness w.r.t. joinability of $\mathbb{U}_{\mathsf{seq}}$ for left-separate DCTRS)**.** *Let $\mathcal{R}$ be a left-separate DCTRS, and let $u, v \in \mathcal{T}$ be two original terms such that $u \downarrow_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} v$, then $u \downarrow_{\mathcal{R}} \mathsf{tb}(v)$.*

Proof. Straightforward from Lemma 1 and Lemma 10. □

Theorem 3 (soundness w.r.t. joinability of $\mathbb{U}_{\mathsf{seq}}$ for left-separate DCTRS)**.** *Let $\mathcal{R}$ be a left-separate DCTRS. Then $\mathbb{U}_{\mathsf{seq}}$ is sound w.r.t. joinability.*

Proof. Straightforward from Lemma 11. □

4.6.3 Confluent CTRSs

Although confluence is a property that we would like to prove using unravelings it is interesting to see that confluence is sufficient to obtain soundness for normal 1-CTRSs, yet not for DCTRS.

For non-sorted 1-CTRSs we immediately obtain unsoundness as we have shown in Example 18. Also for sorted deterministic CTRSs we obtain unsoundness (Example 23).

By adding a rule in which the condition corresponds to the unsound derivation of Example 23

$$A \to B \Leftarrow g(f(a), f(b)) \to^* h(\langle d, k\rangle, \langle d, l\rangle)$$

we even obtain unsoundness w.r.t. joinability for a sorted CTRS: The condition is not satisfied in $\mathcal{R}$, hence $A \not\downarrow_{\mathcal{R}} B$, yet, $A \downarrow_{\mathbb{U}(\mathcal{R})} B$.

In this case the right-hand side of a condition is reducible. In right-stable CTRSs such conditions are not allowed and in fact we proved soundness w.r.t. joinability for these systems in [8]. The proof is similar to the proof of Theorem 2, yet instead of reducibility we prove joinability.

Lemma 12 (technical key result [8, Lemma 6])**.** *Let $\mathcal{R}$ be a confluent, right-stable DCTRS and let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$, $u_1 \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$. Then $\mathsf{tb}(u_m|_{p_m}) \downarrow_{\mathcal{R}} \mathsf{tb}(u_{m+1}|_{p_m})$ for all $m \in \{1, \ldots, n\}$.*

Proof. We prove this result by induction on the length n of A.

For the induction base $|A| = 1$, we either apply an unconditional rule so that $u_1|_{p_1}$ and $u_2|_{p_1}$ are original terms, or an introduction rule so that $\mathsf{tb}(u_2|_{p_1}) = u_1|_{p_1}$.

By the inductive hypothesis, $\mathsf{tb}(u_i|_{p_i}) \downarrow_{\mathcal{R}} \mathsf{tb}(u_{i+1}|_{p_i})$ for $i \in \{1, \ldots, n-1\}$.

For the inductive step we distinguish several cases depending on the rule applied in the last rewrite step on A:

If the rule is an introduction- or switch rule, then $\mathsf{tb}(u_n|_{p_n}) = \mathsf{tb}(u_{n+1}|_{p_n})$.

If it is an original rule, $u_n|_{p_n} = l\sigma$ and $u_{n+1}|_{p_n} = r\sigma$, $l \to r \in \mathcal{R}$ and hence $l\,\mathsf{tb}(\sigma) \to_{\mathcal{R}} r\,\mathsf{tb}(\sigma)$.

If it is an elimination rule, $u_n|_{p_n} = U_k^{\alpha}(t_k\sigma_{k+1}, \overrightarrow{X_k}\sigma_{k+1})$, $u_{n+1}|_{p_n} = r\sigma_{k+1}$, $\mathsf{tb}(u_n|_{p_n}) = l\,\mathsf{tb}(\sigma_{k+1})$ and $\mathsf{tb}(u_{n+1}|_{p_n}) = r\,\mathsf{tb}(\sigma_{k+1})$ where α is the rule $l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$.

We can again extract derivations from A by Lemma 4: $s_i\sigma_i \to^* t_i\sigma_{i+1}$ and $x\sigma_i \to^* x\sigma_{i+1}$ ($x \in X_i$) for all $i \in \{1, \ldots, k\}$. We can apply our inductive hypothesis to these subderivations: $s_i\,\mathsf{tb}(\sigma_i) \downarrow_{\mathcal{R}} t_i\,\mathsf{tb}(\sigma_{i+1})$ and $x\,\mathsf{tb}(\sigma_i) \downarrow_{\mathcal{R}} x\,\mathsf{tb}(\sigma_{i+1})$.

Since $\mathcal{R}$ is confluent this implies $s_i\,\mathsf{tb}(\sigma_{k+1}) \downarrow t_i\,\mathsf{tb}(\sigma_{k+1})$.

By right-stability this implies that there is a τ such that $x\sigma_{k+1} \to^* x\tau$ and $s_i\tau \to^* t_i\tau$ for all $i \in \{1, \ldots, k\}$. Therefore $l\,\mathsf{tb}(\sigma_{k+1}) \to^* l\tau$, $l\tau \to_{\mathcal{R}} r\tau$ and $r\,\mathsf{tb}(\sigma_{k+1}) \to^* r\tau$. □

Hence we obtain soundness w.r.t joinability for confluent, right-stable DCTRSs:

Lemma 13 (soundness w.r.t. joinability of confluent, right-stable DCTRSs [8, Lemma 7])**.** *Let $\mathcal{R}$ be a confluent, right-stable DCTRS, and let $A : u \to^*_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} v$, $u \in \mathcal{T}$ be a derivation in its unraveled TRS, then $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$.*

Proof. Straightforward from Lemma 12 and Lemma 8. □

Theorem 4 (soundness w.r.t. joinability of confluent, right-stable DCTRSs [8, Theorem 8])**.** *Let $\mathcal{R}$ be a confluent, right-stable DCTRS. Then $\mathbb{U}_{\mathsf{seq}}$ is sound.*

Proof. Straightforward from Lemma 13. □

Note, that several methods exist to prove confluence of DCTRSs ([2], [33]).

4.7 Transforming Derivations

Mapping U-terms to original terms is an intuitive approach to prove soundness, yet we can only apply it if we demand very strict syntactic restrictions for conditional rules. For many practically relevant cases we cannot prove soundness using this approach.

We will therefore define properties of derivations that imply soundness and then prove that other derivations can be transformed into derivations that satisfy such properties. We will first use $\mathbb{U}_{\mathsf{opt}}$ because it encodes less information and therefore allows more general results. Yet, we need to consider that $\mathbb{U}_{\mathsf{opt}}$ may give rise to unsoundness if U-terms are erased.

In [14] soundness is proven by showing that so-called balanced reductions are sound. The notion of balanced reductions is not used uniformly in the literature. [29] defines balanced reductions as reductions in which the introduction terms may not contain U-symbols and U-terms are always eliminated. It therefore resembles the membership condition $\in \mathcal{T}$. The definition in [14] is more restrictive and also forbids rewrite steps in variable arguments similar to context-sensitive rewriting. Soundness for left-linear CTRSs is shown in [29] and [14] by converting every derivation of a left-linear CTRSs into a balanced derivation.

The unsound derivation in Example 22 is a balanced reduction (following the definition of [29]). Balanced reductions are therefore not necessarily sound for non-left-linear CTRSs.

We therefore need to define a property that implies soundness of a derivation. One candidate is context-sensitive rewriting: In [31] it is shown that derivations in the sequential unraveling $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ of some DCTRS $\mathcal{R}$ are sound if they use the replacement map $\mu(U) = \{1\}, U \in \mathbb{U}(\mathcal{F}) \setminus \mathcal{F}$ and $\mu(f) = \{1, \ldots, \mathsf{ar}(f)\}, f \in \mathcal{F}$. We could therefore prove soundness for a rewrite sequence by translating it into a context-sensitive rewrite sequence.

Yet, for $\mathbb{U}_{\mathsf{opt}}$, context-sensitive rewriting is not sufficient for soundness. In fact, Example 5 is a counterexample. Even if all U-terms are eliminated we obtain unsoundness:

Example 29 (Unsoundness for context-sensitive rewriting in $\mathbb{U}_{\mathsf{opt}}$)**.** *Consider the following sorted DCTRS and its unraveling:*

$$\mathcal{R} = \left\{ \begin{array}{l} a \begin{array}{l} \nearrow t(c) \\ \searrow t(d) \end{array} \\ f(x,y) \to z \Leftarrow x \to^* t(z) \\ g(x,x) \to h(x,x) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{r} a \begin{array}{l} \nearrow t(c) \\ \searrow t(d) \end{array} \\ f(x,y) \to U_1^\alpha(x) \\ U_1^\alpha(t(z)) \to z \\ g(x,x) \to h(x,x) \end{array} \right\}$$

Consider the terms $f(a,k)$ and $f(a,l)$. Both terms rewrite to the term $U_1^\alpha(a)$ that rewrites to c and d. We therefore can apply the non-linear rule after some rewrite steps:

$$\begin{aligned} g(f(a,k), f(a,l)) &\Vdash g(U_1^\alpha(a), U_1^\alpha(a)) \to h(U_1^\alpha(a), U_1^\alpha(a)) \\ &\Vdash h(U_1^\alpha(t(c)), U_1^\alpha(t(d))) \Vdash h(c,d) \end{aligned}$$

In $\mathcal{R}$, the only common reducts of the terms $f(a,k)$ and $f(a,l)$ are c and d. Therefore, $g(f(a,k), f(a,l))$ only rewrites to $h(c,c)$, $h(d,d)$ and also $g(c,d)$ but not $h(c,d)$.

We also obtain the unsound derivation using context-sensitive rewriting because the U-terms do not contain any variable argument.

For $\mathbb{U}_{\mathsf{seq}}$, proving soundness using context-sensitivity is complex because we cannot translate every rewrite sequence $u \to^* v$ in some $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into a context-sensitive rewrite sequence $u \to^* v$, even if $\mathbb{U}_{\mathsf{seq}}$ is sound:

Example 30 (Reachable U-terms in context-sensitive rewriting)**.** *Consider the following CTRS and its unraveling:*

$$\mathcal{R} = \left\{ \begin{array}{c} a \to b \\ f(x) \to x \Leftarrow x \to^* b \end{array} \right\} \quad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{c} a \to b \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(b,x) \to x \end{array} \right\}$$

Consider the following derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$*:*

$$f(a) \to U_1^\alpha(a,a) \to U_1^\alpha(a,b)$$

The rewrite step $U_1^\alpha(a,a) \to U_1^\alpha(a,b)$ *does not satisfy the condition of context-sensitivity for the replacement map* $\mu(U_1^\alpha) = \{1\}$*. In fact, there is no context-sensitive derivation* $f(a) \hookrightarrow_\mu^* U_1^\alpha(a,b)$*.*

We will therefore define a different rewrite strategy for our soundness proofs.

4.7.1 U-Eagerness

In [14] soundness of left-linear normal 1-CTRSs is shown using balanced reductions. We introduce a similar but more strict strategy: If we introduce a U-term in a derivation, then all succeeding rewrite steps must be (not strictly) below this U-term until it is eliminated. We even forbid rewrite steps in parallel positions. We call such derivations *U-eager*:

Definition 16 (U-eagerness)**.** *A derivation* $A : u_1 \to_{p_1} u_2 \to_{p_2} \cdots \to_{p_n} u_{n+1}$ *in some unraveling* $\mathbb{U}(\mathcal{R})$ *is U-eager if all U-terms are immediately rewritten, i.e., for all U-terms* $u_m|_p$*,* $p \leq p_m$ *(*$m \in \{1, \ldots, n+1\}$*).*

This definition implies two important facts:

- If a derivation $u \to^*_{\mathbb{U}(\mathcal{R})} v$ is U-eager and u, v are original terms, then all U-terms are uniquely eliminated in this derivation.
- If a derivation $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} w$ is U-eager and u, v, w are original terms, then $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$ and $v \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} w$ are U-eager.

In the following, we will refer to rewrite steps above U-terms as *non-U-eager rewrite steps*.

U-eagerness allows us to easily extract rewrite sequences from the variable and conditional arguments because they must be connected. Since U-eagerness implies that introduction terms, switch terms and elimination terms do not contain U-terms we can use an inductive argument to prove soundness for sorted CTRSs (for non-sorted CTRSs Example 18 is a counterexample).

Lemma 14 (soundness of U-eagerness)**.** *Let $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$, $u, v \in \mathcal{T}$ be a U-eager derivation in the unraveling of a sorted DCTRS $\mathcal{R}$, then $u \to^*_{\mathcal{R}} v$.*

In the following, we assume w.l.o.g. $\mathcal{VR}an(\sigma) \cap \mathcal{D}om(\sigma) = \emptyset$ for all substitutions σ.

Proof. Let $A : u_1 \to_{p_1} u_2 \to_{p_2} \cdots \to_{p_n} u_{n+1}$ ($u_1, u_{n+1} \in \mathcal{T}$) be a U-eager derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.

We prove by induction on the length n of the derivation that $u_1 \to^*_{\mathcal{R}} u_{n+1}$.

If $n = 1$, then the applied rule in $u_1 \to_{l\to r} u_2$ is an unconditional rule in $\mathcal{R}$ so that also $u_1 \to_{\mathcal{R}} u_2$.

For the inductive step, we distinguish two cases:

First, if there is some u_m ($m \in \{2, \ldots, n\}$) such that $u_m \in \mathcal{T}$, then the derivations $u_1 \to^+ u_m$ and $u_m \to^+ u_{n+1}$ are U-eager, and by the inductive hypothesis $u_1 \to^*_{\mathcal{R}} u_m$ and $u_m \to^*_{\mathcal{R}} u_{n+1}$.

Second, if there is no such u_m, the first rewrite step in A is the introduction step of some rule $\alpha : l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, the last rewrite step in A is the corresponding elimination step, $p_1 = p_n$ and all rewrite steps are below p_1, $p_1 \leq p_m$ for all $m \in \{1, \ldots, n\}$.

Let the introduction term be $l\sigma_1 = u_1|_p$ and the elimination term $r\sigma_{k+1} = u_{n+1}|_p$ ($p = p_1 = p_n$).

By Lemma 4 we obtain derivations $s_1\sigma_1 \to^* t_1\sigma_2, \ldots, s_k\sigma_k \to^* t_k\sigma_{k+1}$ and $x\sigma_i \to^* x\sigma_{i+1}$ for all $x \in \mathcal{V}ar(\alpha)$ and $i \in \{1, \ldots, k\}$. The range of the substitutions $\sigma_1, \ldots, \sigma_{k+1}$ only contains original terms because they are the matchers in switch terms and A is U-eager. Furthermore, the length of these derivations is less than n. Therefore, the inductive hypothesis holds for all these subderivations.

We now combine these substitutions in the following way:

$$\sigma_{\mathit{left}} = \{x \mapsto x\sigma_i \mid i \text{ is the minimum } i \text{ such that } x \in \mathcal{V}ar(s_i)\}$$
$$\sigma_{\mathit{right}} = \{x \mapsto x\sigma_{i+1} \mid i \text{ is the maximum } i \text{ such that } x \in \mathcal{V}ar(t_i)\}$$

These two substitutions give rise to the following derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ and by the inductive hypothesis also in $\mathcal{R}$: $x\sigma_i \to^* x\sigma_{\mathit{right}}$ for all $\{i \mid x \in \mathcal{V}ar(t_j), i \leq j\}$ and $x\sigma_{\mathit{left}} \to^* x\sigma_i$ for all $\{i \mid x \in \mathcal{V}ar(s_j), j \leq i\}$. Furthermore, by the definition of sortedness $x\sigma_{\mathit{left}} = x\sigma_{\mathit{right}}$ for all $x \in \mathcal{D}om(\sigma_{\mathit{left}}) \cap \mathcal{D}om(\sigma_{\mathit{right}})$.[2]

The substitution $\sigma = \sigma_{\mathit{left}} \circ \sigma_{\mathit{right}} \circ \sigma_1$ is defined as the combination of these two substitutions and the matcher σ_1 of the introduction term.

We now obtain the following derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ and by the inductive hypothesis also in $\mathcal{R}$: $l\sigma_1 \to^* l\sigma$ and $r\sigma \to^* r\sigma_{k+1}$. Furthermore, $s_i\sigma \to^* s_i\sigma_i$ and $t_i\sigma_{i+1} \to^* t_i\sigma$. The latter implies that also $s_i\sigma \to^* t_i\sigma$.

Therefore, the conditions are satisfied for σ so that $l\sigma \to_{\mathcal{R}} r\sigma$, and finally

$$l\sigma_1 \to^*_{\mathcal{R}} l\sigma \to_{\mathcal{R}} r\sigma \to^*_{\mathcal{R}} r\sigma_{k+1}.$$

□

[2] In the non-sorted CTRS of Example 18, $x\sigma_{\mathit{left}} = a$ and $x\sigma_{\mathit{right}} = b$.

The following example illustrates the previous soundness proof:

Example 31. *Consider the following sorted CTRS and its unraveling.*

$$\mathcal{R} = \left\{ \begin{array}{c} a \nearrow c \\ a \searrow d \\ f(x,y) \to z \Leftarrow c \to^* x, y \to^* z \end{array} \right\} \qquad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{c} a \nearrow c \\ a \searrow d \\ f(x,y) \to U_1^\alpha(c,x,y) \\ U_1^\alpha(x,x,y) \to U_2^\alpha(y) \\ U_2^\alpha(z) \to z \end{array} \right\}$$

We obtain the following U-eager derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$:

$$f(a,a) \to U_1^\alpha(c,a,a) \to U_1^\alpha(c,c,a) \to U_2^\alpha(a) \to U_2^\alpha(d) \to d$$

The substitutions used in the induction step of the proof of Lemma 14 are as follows:

$$\begin{aligned} \sigma_1 &= \{x \mapsto a, y \mapsto a\} \\ \sigma_2 &= \{x \mapsto c, y \mapsto a\} \\ \sigma_3 &= \{z \mapsto d\} \\ \sigma_{left} &= \{y \mapsto a\} \\ \sigma_{right} &= \{x \mapsto c, z \mapsto d\} \\ \sigma &= \{x \mapsto c, y \mapsto a, z \mapsto d\} \end{aligned}$$

Observe that the conditions are not satisfied for σ_1 because $s_1\sigma_1 = c$, $t_1\sigma_1 = a$ but $c \not\to^*_{\mathcal{R}} a$. Yet, $s_1\sigma = c$, $t_1\sigma = c$ and $c \to^*_{\mathcal{R}} c$. Furthermore $s_2\sigma = a$, $t_2\sigma = d$ and $a \to^*_{\mathcal{R}} d$, hence the conditions are satisfied for σ and we obtain the following derivation in $\mathcal{R}$:

$$f(a,a) \to f(c,a) \to a \to d$$

4.7.2 Membership Condition

In [18] it is shown that a combination of rewriting using the membership condition $\in \mathcal{T}$ and context-sensitive rewriting is sufficient to obtain soundness for $\mathbb{U}_{\mathsf{opt}}$. [31] proves that context-sensitive rewriting is sufficient for $\mathbb{U}_{\mathsf{seq}}$.

Yet, we do not obtain soundness for context-sensitive rewriting for $\mathbb{U}_{\mathsf{opt}}$. The unsound derivation of Example 5 is a counterexample. The membership condition $\in \mathcal{T}$ is also not sufficient in general: By the definition of U-eagerness, every U-eager derivation also satisfies the membership condition, but U-eagerness only implies soundness for sorted CTRSs.

Nevertheless, for sorted CTRSs we can prove soundness for derivations that satisfy the membership condition $\in \mathcal{T}$. Such derivations allow rewrite steps in positions that are parallel to U-terms. The definition of U-eagerness dose not allow such rewrite steps. Yet, the next lemma shows that every membership derivation can be transformed into a U-eager derivation by grouping rewrite steps below the same U-term:

Lemma 15. *Let $\mathcal{R}$ be a DCTRS $\mathcal{R}$ and $A : u \hookrightarrow^*_{\in\mathcal{T}} v$, $u, v \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. Then, there is a U-eager derivation $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$.*

Proof. Let A be the derivation $u_1 \hookrightarrow_{\in\mathcal{T},p_1} u_2 \cdots \hookrightarrow_{\in\mathcal{T},p_n} u_{n+1}$ where $u_1, u_{n+1} \in \mathcal{T}$. Observe that the membership condition $\in \mathcal{T}$ implies that all $u_m|_{p_m.q}$ ($q \in \mathcal{P}os(u_m|_{p_m}) \setminus \{\epsilon\}$) are original terms, and furthermore that all U-terms are (uniquely) eliminated.

We show by induction on the number of non-U-eager rewrite steps that we can remove all non-U-eager rewrite steps:

If A does not contain any non-U-eager rewrite step, our results holds vacuously.

Otherwise, let first m be the smallest value and second p be the innermost position such that $u_m|_p$ is a U-term and $p < p_m$ or $p \parallel p_m$, i.e., the mth rewrite step is not U-eager. This implies that for all U-terms $u_m|_q$, $q \leq p$.

Let j be the smallest value such that $p \leq p_{m+j}$, i.e., the $m+j$th rewrite step in A is the next rewrite step in which a descendant of $u_m|_p$ is rewritten. We obtain following derivation:

$$u_1 \to^* u_m \to_{p_{m+j}} u_m[u_{m+j+1}|_p]_p$$

This derivation is U-eager because $p \leq p_{m+j}$ and for all U-terms $u_m|_q$, $q \leq p \leq p_{m+j}$.

By the membership condition, p is parallel to the positions $p_{m+1}, \ldots, p_{m+j-1}$. We therefore obtain the following derivation

$$u_m[u_{m+j+1}|_p]_p \to_{p_m} u_{m+1}[u_{m+j+1}|_p]_p \cdots \to_{p_{m+j-1}} u_{m+j}[u_{m+j+1}]_p$$

where $u_{m+j}[u_{m+j+1}]_p = u_{m+j+1}$.

By combining this derivation with the U-eager derivation we obtain

$$\begin{aligned} u_1 \to^* u_m \to_{p_{m+j}} u_m[u_{m+j+1}]_p \to_{p_m} u_{m+1}[u_{m+j+1}]_p \cdots \\ \to_{p_{m+j-1}} u_{m+j}[u_{m+j-1}]_p = u_{m+j-1} \to^* u_{n+1} \end{aligned}$$

that contains less non-U-eager rewrite steps than A. Therefore we can apply our inductive hypothesis. □

Since U-eager derivations are sound we also obtain soundness for derivations that satisfy the membership condition:

Lemma 16 (soundness for the membership condition)**.** *Let $u \hookrightarrow^*_{\in\mathcal{T},\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$, $u, v \in \mathcal{T}$ be a derivation in the unraveled TRS of some sorted DCTRS $\mathcal{R}$, then $u \to^*_{\mathcal{R}} v$.*

Proof. By Lemma 15 there is a U-eager derivation $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$. Lemma 14 yields $u \to^*_{\mathcal{R}} v$. □

Theorem 5 (soundness for the membership condition)**.** $\mathbb{U}_{\mathsf{opt}}$ *is sound for derivations using the membership condition for sorted DCTRSs.*

Proof. Straightforward from Lemma 16. □

4.7.3 U-Right-Linear Derivations

In U-eager derivations, we must rewrite U-terms immediately. We do not allow any rewrite steps outside of U-terms. In derivations that satisfy the membership condition $\in \mathcal{T}$ we weaken this requirement and allow rewrite steps in parallel positions.

In the following we show that we can weaken this requirement even further and prove soundness of derivations in which all U-terms are eliminated and have at most one one-step descendant. This rewriting strategy is equivalent to rewriting using another membership condition

$$l \to r \Leftarrow \bigwedge_{x \in \mathcal{V}ar(l) \wedge |l|_x > 1} x \in \mathcal{T}.$$

In such derivations we can trace every introduction term to a unique elimination term. By inductively extracting these derivations and recombining them we obtain a derivation that satisfies the membership condition.

We will refer to derivations in which all U-terms have at most one one-step descendant as *U-right-linear derivations* in the following.

We first prove that we can split up derivations that satisfy the membership condition. This will be necessary in the induction step of our key result.

Lemma 17. *Let $\mathcal{R}$ be a sorted DCTRS and $A : u_1 \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$, $u_{n+1} \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. Let furthermore $u_1|_p$ be a U-term. Then, there are derivations $u_1|_p \hookrightarrow^*_{\in\mathcal{T}} r\sigma$ and $u_1[r\sigma]_p \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$ in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ where $\mathsf{te}(A, 1, p) = \{(m, p)\}$ and $r\sigma = u_{m+1}|_p$ is the elimination term of $u_1|_p$.*

Proof. We prove this lemma by induction on m:

If $m = 1$, then $u_2 = u_1[r\sigma]_p$ and $p_1 = p$. $u_1|_p \to_\epsilon u_2|_p$ and $u_2 \to^* u_{n+1}$ satisfy the membership condition by assumption.

Otherwise, $u_2|_p$ is a U-term and $\mathsf{te}(A, 2, p) = \mathsf{te}(A, 1, p) = \{(m, p)\}$. By the inductive hypothesis, there are two derivations $u_2|p \hookrightarrow^*_{\in\mathcal{T}} r\sigma$ and $u_2[r\sigma]_p \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$. Since the combination of two rewrite sequences that satisfy the membership condition also satisfies the membership condition it remains to show that $u_1[r\sigma]_p \hookrightarrow^*_{\in\mathcal{T}} u_2[r\sigma]_p$ and $u_1|_p \hookrightarrow^*_{\in\mathcal{T}} u_2|_p$.

p_1 is the position of the redex in u_1. We distinguish three cases based on the position p_1 and p:

The case $p_1 < p$ contradicts the membership condition because $u_1|_p$ is a U-term.

If $p_1 \parallel p$, then $u_1|_p = u_2|_p$. Since $u_1|_{p_1} \to u_2|_{p_1}$ satisfies the membership condition $\in \mathcal{T}$, so does $u_1[r\sigma]_p \to_{p_1} u_2[r\sigma]_p$.

Finally, assume $p \leq p_1$. In this case, $u_1[r\sigma]_p = u_2[r\sigma]_p$, therefore $u_1[r\sigma]_p \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$. Since $u_1|_p \to u_2|_p$ also satisfies the membership condition $\in \mathcal{T}$ so does $u_1|_p \to^* r\sigma$. □

We can transform U-right-linear derivations into derivations that satisfy the membership condition by repeatedly applying the previous lemma:

Lemma 18. *Let $A : u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$, $v \in \mathcal{T}$ be a derivation in the unraveled TRS of a sorted DCTRS $\mathcal{R}$, such that all U-terms are eliminated and have at most one one-step descendant. Then, there is a derivation $u \hookrightarrow^*_{\in\mathcal{T}} v$ in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.*

Proof. Let A be the derivation $u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ where $u_{n+1} \in \mathcal{T}$ and u_1 is a possibly mixed term. We show this lemma by induction on the length n of the derivation.

If n is 1, then $u_2 \in \mathcal{T}$. Since all U-terms in A are eliminated u_1 can only contain a U-term at position p_1. Therefore, A satisfies the membership condition.

Otherwise by the inductive hypothesis there is a derivation $A_{\in\mathcal{T}} : u_2 \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$ in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.

We prove that we obtain a derivation $u_1 \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$ by induction on the number of U-terms $u_2|_q$ such that $p_1 < q$.

If there are no such U-terms then $u_1|_{p_1} \to u_2|_{p_1}$ satisfies the membership condition because all U-terms in A (and therefore also those in $u_1|_{p_1}$) are eliminated.

Otherwise let $u_2|_q$ be a U-term of the conditional rule $l \to r \Leftarrow c$ such that there is no other U-term below $u_2|_q$.

Since $A_{\in\mathcal{T}}$ satisfies the membership condition $u_2|_q$ has the unique elimination term $r\tau$ in $A_{\in\mathcal{T}}$. Observe that the matcher τ does not contain any U-terms because the elimination step of $u_2|_q$ in $A_{\in\mathcal{T}}$ satisfies the membership condition. $r\tau$ is therefore an original term.

We can apply Lemma 17 to $A_{\in\mathcal{T}}$ and obtain derivations $u_2[r\tau]_q \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$ and $u_2|_q \hookrightarrow^*_{\in\mathcal{T}} r\tau$.

Let Q be the positions of all one-step ancestors of $u_2|_q$ in u_1.

Since all U-terms have at most one one-step descendant, $u_2|_q$ is the only one-step descendant of $u_1|_{q'}$ for all $q' \in Q$. Therefore also $u_1[r\tau]_Q \to_{p_1} u_2[r\tau]_q$.

In the rewrite step $u_1[r\tau]_Q \to_{p_1} u_2[r\tau]_q$ there is one U-term less below $u_2|_{p_1}$ than in $u_1|_{p_1} \to u_2|_{p_1}$. Hence, we can apply the inductive hypothesis to $u_1[r\tau]_Q \to_{p_1} u_2[r\tau]_q \to^* u_{n+1}$ and obtain $u_1[r\tau]_Q \hookrightarrow^*_{\in\mathcal{T}} u_{n+1}$.

Finally, since $u_2|_q \hookrightarrow^*_{\in\mathcal{T}} r\tau$ and $u_1|_{q'} = u_2|_q$ for all $q' \in Q$, $u_1 \twoheadleftarrow^*_{\in\mathcal{T},Q} u_1[r\tau]_Q$. □

In the following example we show how to construct a derivation that satisfies the membership condition out of a U-right-linear derivation:

Example 32. *Consider the following CTRS and its unraveled TRS:*

$$\mathcal{R} = \left\{ \begin{array}{c} a \to b \\ f(x) \to A \Leftarrow x \to^* b \\ g(x,x) \to h(x) \\ h(x) \to i(x) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{c} a \to b \\ f(x) \to U_1^\alpha(x) \\ U_1^\alpha(b) \to A \\ g(x,x) \to h(x) \\ h(x) \to i(x) \end{array} \right\}$$

In the following derivation all U-terms are eliminated and have at most one one-step descendant:

$$g(f(a),f(a)) \Vdash g(U_1^\alpha(a),U_1^\alpha(a)) \to h(U_1^\alpha(a)) \to h(U_1^\alpha(b)) \to h(A) \to i(A)$$

The last rewrite step in the derivation that does not satisfy the membership condition is $g(U_1^\alpha(a),U_1^\alpha(a)) \to h(U_1^\alpha(a))$.

We can therefore apply Lemma 17 to the rest of the derivation $h(U_1^\alpha(a)) \to h(U_1^\alpha(b)) \to h(A) \to i(A)$ *and split this derivation into two derivations* $U_1^\alpha(a) \to U_1^\alpha(b) \to A$ *and* $h(A) \to$

$i(A)$. *The right-hand side of the conditional rule is A so that we obtain $g(A, A) \to h(A)$ (this corresponds to the rewrite step $u_1[r\tau]_Q \Rrightarrow u_2[r\sigma]_q$ in the proof of Lemma 18).*

The rewrite step $g(A, A) \to h(A)$ satisfies the membership condition. Finally we append the other extracted derivation:

$$g(U_1^\alpha(a), U_1^\alpha(a)) \Rrightarrow g(U_1^\alpha(b), U_1^\alpha(b)) \Rrightarrow g(A, A) \to h(A) \to i(A)$$

Using Lemma 18 and our soundness result for membership rewriting we now can prove soundness of $\mathbb{U}_{\mathsf{opt}}$ for U-right-linear derivations:

Lemma 19 (soundness for U-right-linear derivations)**.** *Let $u \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v$, $u, v \in \mathcal{T}$ be a derivation such that all U-terms are eliminated and all U-terms have at most one-step descendant. Then, $u \to^*_{\mathcal{R}} v$.*

Proof. By Lemma 18 we can transform all such derivations into a membership derivation. By Lemma 16 all such derivations are sound. □

Lemma 19 implies soundness of $\mathbb{U}_{\mathsf{opt}}$-non-erasing and $\mathbb{U}_{\mathsf{opt}}$-right-linear DCTRSs because in such systems all derivations are U-right-linear. This result has already been shown in [25]. Yet, in our result we focus on properties of rewrite sequences instead of syntactic properties of the DCTRS. We can therefore use Lemma 19 to prove soundness also for other CTRSs.

4.7.4 Erased and Eliminated U-Terms w.r.t. Junk Terms

The unravelings $\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$ both sequentially encode the conditions of conditional rules. Yet, in $\mathbb{U}_{\mathsf{opt}}$, we only encode those variable arguments that we "need" in potential further rewrite steps. Although this more efficient way of encoding is more efficient, it causes unsoundness if U-terms are erased (see Example 5).

In the soundness result for U-right-linear derivations we therefore can only prove soundness for derivations in which all U-terms are eliminated (for U-eager derivations and derivations that satisfy the membership condition this is always satisfied). This is an important restriction because in a rewrite engine that uses $\mathbb{U}_{\mathsf{opt}}$ we need to know that the condition is satisfiable before introducing the U-term itself.

In $\mathbb{U}_{\mathsf{seq}}$, we encode all variable arguments of the left-hand side. Therefore the backtranslation $\mathtt{tb}$ is always properly defined. We therefore can translate intermediate results of conditional evaluations even if the condition is not satisfied.

Yet, compared to $\mathbb{U}_{\mathsf{opt}}$ we now encode "too much". If we prove soundness in a derivation in which a U-term is erased, it does not matter whether the condition is satisfiable or not. We can therefore ignore rewrite steps in the conditional argument of this erased U-term. On the other hand, if a U-term is eliminated, we only need the conditional argument and those variable arguments that we also encode in $\mathbb{U}_{\mathsf{opt}}$ because we would not translate the U-term to an original term.

This distinction of necessary arguments and unnecessary arguments in U-terms is essential for the following results because we translate rewrite sequences in an unraveled TRS into conditional rewrite sequences. For this purpose we must distinguish erased U-terms (that are replaced

by the corresponding left-hand side) from eliminated U-terms (the rewrite sequence that is extracted from the conditional argument is replaced by rewrite steps in the conditions). Yet, for $\mathbb{U}_{\mathsf{seq}}$ we obtain the paradox situation that a U-term that is eliminated in an unnecessary argument might actually be erased.

Example 33. *Consider the following CTRS and its unraveled TRS using $\mathbb{U}_{\mathsf{seq}}$:*

$$\mathcal{R} = \left\{ \begin{array}{l} f(x) \to x \Leftarrow x \to^* a \\ g(x) \to C \Leftarrow A \to^* B \\ A \to B \end{array} \right\} \qquad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{r} f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(a,x) \to x \\ g(x) \to U_1^\beta(A,x) \\ U_1^\beta(B,x) \to C \\ A \to B \end{array} \right\}$$

In the following derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$, all U-terms are eliminated:

$$g(f(a)) \to g(U_1^\alpha(a,a)) \to U_1^\beta(A, U_1^\alpha(a,a)) \to U_1^\beta(A,a) \to U_1^\beta(B,a) \to C$$

We apply an elimination step to the U-term that is introduced by the transformed g-rule. Hence, we extract the subderivation $A \to B$ from the conditional argument. Replacing the elimination step and the rewrite step in the conditional argument by a conditional rewrite step therefore yields the following derivation:

$$g(f(a)) \to g(U_1^\alpha(a,a)) \xrightarrow{\Leftarrow A \to^* B}_\beta C$$

Although the second U-term $U_1^\alpha(a,a)$ was eliminated in the initial derivation, it now is erased. Since $U_1^\alpha(a,a)$ does not have any one-step descendants anymore it is actually erased and not eliminated.

We therefore replace this U-term by its corresponding introduction term using the current variable bindings. Hence, translating the derivation into the original CTRS yields the following derivation:

$$g(f(a)) \to_{\mathcal{R}} C$$

Instead of translating derivations in some $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into derivations in $\mathcal{R}$ we rather translate them into derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ so that we can apply the soundness result of Lemma 19.

We cannot prove soundness for derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ that contain erased U-terms, therefore we need to replace these U-terms by original terms. Yet again we need to analyze which terms are eliminated or erased in such unnecessary arguments.

Before transforming the derivation we therefore mark all terms in such unnecessary arguments as *junk terms*. Such junk terms will be ignored in our translation. After transforming a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ in which all U-terms are eliminated there are no unnecessary arguments and therefore no junk terms.

In order to analyze which U-term is actually erased and eliminated we need to distinguish junk terms and non-junk-terms. Since therefore the notion of junk terms depends on whether a U-term is eliminated or erased, we adapt the notion of erased and eliminated terms.

The rewrite step $g(U_1^\alpha(a,a)) \to U_1^\beta(A, U_1^\alpha(a,a))$ puts a U-term into an unnecessary argument in a U-term. Therefore, $U_1^\alpha(a,a)$ is erased w.r.t. junk terms term even though it is eliminated in the initial derivation.

The formal definition of such junk terms is more complicated because the notions of junk terms, eliminated and erased U-terms are self-referring. For the sake of readability, we will refer to terms $u_m|_p$ as junk terms although we refer to an index and position in some derivation A.

Definition 17 (junk terms)**.** *Let $A : u_1 \to^*_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} u_{n+1}$ be a derivation in some unraveled CTRS. We assume that all U-terms in u_{n+1} are erased w.r.t. junk terms.*

Let $u_m|_p$ be a U-term $U_j^\alpha(v, \overrightarrow{X_j}\sigma)$. A term $u_m|_q$ ($p < q$) is a junk term if

- *$u_m|_p$ is erased w.r.t. junk terms and $u_m|_q$ is below the conditional argument, $p.1 \leq q$.*
- *$u_m|_p$ is erased w.r.t. junk terms and $u_m|_q$ is below an extra variable argument, i.e., if $p.i \leq q$ for $i \in \mathbb{N}$ then $\overrightarrow{X_j}[i-1] \notin \mathcal{V}ar(l)$.*
- *$u_m|_p$ is eliminated w.r.t. junk terms and $u_m|_q$ is inside a variable argument that is not needed for the evaluation of the conditions, i.e., if $p.i \leq q$ for $i \in \mathbb{N}$ then $\overrightarrow{X_j}[i-1] \notin \mathcal{V}ar(t_j, s_{j+1}, t_{j+1}, \ldots, t_k, r)$*

A one-step descendant $u_{m+1}|_{q'}$ of q is a non-junk one-step descendant *if $u_{m+1}|_{q'}$ is not a junk term.*

The U-term $u_m|_p$ is erased w.r.t. junk terms *if it has no non-junk one-step descendants, or if all its non-junk one-step descendants are also erased w.r.t. junk terms.*

It is eliminated w.r.t. junk terms *if it has one non-junk one-step descendant that is not a U-term, or if (at least) one one-step descendant is eliminated w.r.t. junk terms.*

It is fully eliminated w.r.t. junk terms *if it has one non-junk one-step descendant $(m+1, p')$ that is not a U-term, or if it all its non-junk one-step descendants are fully eliminated w.r.t. junk terms.*

By applying this definition to the last term in a derivation and iteratively progressing to the first term we mark all U-terms as either erased or eliminated w.r.t. junk terms and identify all junk terms. Observe that all descendants and subterms of junk terms are also junk terms. Furthermore, all junk terms are subterms of U-terms.

In the remainder of this section, we will sometimes use the notion "w.r.t. junk terms" implicitly.

We will in the following attempt to translate rewrite sequences in some $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into rewrite sequences in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ of which we already have proven soundness. For this purpose we will replace erased (w.r.t. junk terms) U-terms by the corresponding left-hand side of the conditional rule. All other U-terms are replaced by the corresponding U-term in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. In this process, all junk terms are deleted. In the resulting derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ all U-terms are eliminated, therefore it contains no no junk terms.

Yet, we cannot translate all derivations in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$. There are derivations in which we cannot replace erased U-terms with the corresponding left-hand side because one of their ancestors is eliminated w.r.t. junk terms:

Example 34. *Consider the following CTRS $\mathcal{R}$ and its unraveled TRS*

$$\mathcal{R} = \left\{ \begin{array}{c} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ g(x,x) \to h(x,x) \\ f(x) \to x \Leftarrow x \to^* c \end{array} \right\} \qquad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{c} a \to c \\ \diagdown\!\!\!\diagup \\ b \to d \\ g(x,x) \to h(x,x) \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(c,x) \to x \end{array} \right\}$$

Now consider the following derivation:

$$\begin{aligned} g(f(a),f(b)) &\to^* g(U_1^\alpha(a,a),U_1^\alpha(b,b)) \to^* g(U_1^\alpha(c,d),U_1^\alpha(c,d)) \\ &\to h(U_1^\alpha(c,d),U_1^\alpha(c,d)) \to h(d,U_1^\alpha(c,d)) \end{aligned}$$

In the last term of the derivation $h(d,U_1^\alpha(c,d))$, the U-term is not eliminated. We treat it as an erased U-term because it has no one-step descendants. In order to translate this derivation into a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ without erased U-terms we therefore replace this U-term by the corresponding left-hand side of the rule $f(d)$.

In order to apply the non-linear rule $g(x,x) \to h(x,x)$ we need a common reduct u of $f(a)$ and $f(b)$ such that u rewrites to $f(d)$ and d, but there is no such term in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. Hence, there is no derivation $g(f(a),f(b)) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} h(d,f(d))$.

In the term $g(U_1^\alpha(c,d),U_1^\alpha(c,d))$ both U-terms have one one-step descendant that is eliminated (w.r.t. junk terms) and one that is erased (w.r.t. junk terms). They are therefore not fully eliminated. In this case we cannot translate derivations in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.

Hence, we can only translate derivations in which all eliminated U-terms are also fully eliminated.

In order to translate these derivations we need to map terms from the signature of $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into the signature of $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. We will replace fully eliminated (w.r.t. junk terms) U-terms by the corresponding U-term in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ and translate erased U-terms into the corresponding left-hand side of the conditional rule:

Definition 18 ($\mathsf{t}_{\mathsf{opt}}$)**.** *Let A be a derivation $u_1 \to^* u_{n+1}$ in some $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all U-term are either erased or fully eliminated w.r.t. junk terms. Then the mapping $\mathsf{t}_{\mathsf{opt}}$ is defined as follows:*

$$\mathsf{t}_{\mathsf{opt}}(A,m,p) = \begin{cases} x & \text{if } u_m|_p = x \text{ is a variable} \\ f(\mathsf{t}_{\mathsf{opt}}(A,m,p.1),\ldots,\mathsf{t}_{\mathsf{opt}}(A,m,p.\mathsf{ar}(f)) & \text{if } \mathsf{root}(u_m|_p) = f \text{ and } f \in \mathcal{F} \\ l\sigma_{\mathsf{opt}} & \text{if } u_m|_p = U_j^{\mathsf{seq},\alpha}(v,\overrightarrow{X_j^{\mathsf{seq}}}\sigma) \text{ is erased w.r.t. junk terms where } l \text{ is the left-hand side of } \alpha \\ U_j^{\mathsf{opt},\alpha}(\mathsf{t}_{\mathsf{opt}}(A,m,p.1),\overrightarrow{X_j^{\mathsf{opt}}}\sigma_{\mathsf{opt}}) & \text{if } u_m|_p = U_i^{\mathsf{seq},\alpha}(v,\overrightarrow{X_j^{\mathsf{seq}}}\sigma) \text{ is eliminated w.r.t. junk terms} \end{cases}$$

where $\sigma_{\mathsf{opt}} = \left\{x \mapsto \mathsf{t}_{\mathsf{opt}}(A, m, p.(i+1)) \mid x \in X_j^{\mathsf{opt}}, x = \overrightarrow{X_j^{\mathsf{seq}}}[i]\right\}$*,* α *is the rule* $l \to r \Leftarrow c$ *and* X_j^{seq} *and* X_j^{opt} *are the variable sets defined in the* i*th rewrite rule in* $\mathbb{U}_{\mathsf{seq}}(\alpha)$ *and* $\mathbb{U}_{\mathsf{opt}}(\alpha)$*.*

Example 35. *Consider the CTRS of Example 33 and its unraveled TRS using* $\mathbb{U}_{\mathsf{seq}}$ *and* $\mathbb{U}_{\mathsf{opt}}$*:*

$$\mathcal{R} = \left\{\begin{array}{l} f(x) \to x \Leftarrow x \to^* a \\ g(x) \to C \Leftarrow A \to^* B \\ A \to B \end{array}\right\}$$

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{\begin{array}{r} f(x) \to U_1^{\mathsf{seq},\alpha}(x,x) \\ U_1^{\mathsf{seq},\alpha}(a,x) \to x \\ g(x) \to U_1^{\mathsf{seq},\beta}(A,x) \\ U_1^{\mathsf{seq},\beta}(B,x) \to C \\ A \to B \end{array}\right\} \quad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{\begin{array}{r} f(x) \to U_1^{\mathsf{opt},\alpha}(x,x) \\ U_1^{\mathsf{opt},\alpha}(a,x) \to x \\ g(x) \to U_1^{\mathsf{opt},\beta}(A) \\ U_1^{\mathsf{opt},\beta}(B) \to C \\ A \to B \end{array}\right\}$$

We translate the following derivation to $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ *using* $\mathsf{t}_{\mathsf{opt}}$*:*

$$g(f(a)) \to g(U_1^{\alpha}(a,a)) \to U_1^{\beta}(A, U_1^{\alpha}(a,a)) \to U_1^{\beta}(A,a) \to U_1^{\beta}(B,a) \to C$$

First we mark all junk terms. Since all U_1^{β}*-rooted terms are eliminated w.r.t. junk terms, all terms in their variable arguments are junk terms. Hence, the* U_1^{α}*-rooted terms are erased w.r.t. junk terms so that its conditional argument contains junk terms.*

The rewrite sequence is therefore translated into the following derivation (junk terms are underlined):

$$g(f(a)) \to g(\underbrace{U_1^{\mathsf{seq},\alpha}(\underline{a},a)}_{f(a)}) \to \underbrace{U_1^{\mathsf{seq},\beta}(A,\underline{U_1^{\alpha}(a,a)})}_{U_1^{\mathsf{opt},\beta}(A)} \to \underbrace{U_1^{\mathsf{seq},\beta}(A,\underline{a})}_{U_1^{\mathsf{opt},\beta}(A)} \to \underbrace{U_1^{\mathsf{seq},\beta}(B,\underline{a})}_{U_1^{\mathsf{opt},\beta}(B)} \to C$$

Since all U-terms in the last term of a derivation are considered to be erased they are replaced by the corresponding left-hand side of the conditional rule. Therefore, we obtain the following interesting connection between $\mathsf{t}_{\mathsf{opt}}$ and tb:

Lemma 20 ($\mathsf{t}_{\mathsf{opt}}$ and tb)**.** *Let* $\mathcal{R}$ *be some CTRS and* $A : u_1 \to^*_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} u_{n+1}$ *be a derivation such that all U-terms are either fully eliminated or erased. Then,* $\mathsf{t}_{\mathsf{opt}}(A, n+1, \epsilon) = \mathsf{tb}(u_{n+1})$*.*

Proof. Since all U-terms in the last term of a derivation are erased (w.r.t. junk terms), all U-terms are replaced by the corresponding left-hand side of the conditional rule which corresponds to the definition of tb. □

In the following we prove that $\mathsf{t}_{\mathsf{opt}}$ is an appropriate way to translate derivations from $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into derivations in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$. Since we need to translate U-terms from $\mathbb{U}_{\mathsf{seq}}$ into $\mathbb{U}_{\mathsf{opt}}$ these proofs are syntactically challenging.

First we prove that terms that match the same variable in a rule application are mapped to syntactically equal terms in $\mathsf{t}_{\mathsf{opt}}$. This is not trivially satisfied because $\mathsf{t}_{\mathsf{opt}}$ depends on the derivation and not on the syntactic structure of terms as tb or tf. This property is only satisfied if all U-terms are fully eliminated or erased (w.r.t. junk terms).

Lemma 21. *Let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all U-terms are either erased or fully eliminated w.r.t. junk terms. Let $u_m \to_{l\to r} u_{m+1}$ be a rewrite step in A such that $u_m|_{p_m}$ is not a junk term. Let $x \in \mathcal{V}ar(l)$ and $q_1, q_2 \in \mathcal{VP}os(l)$ such that $l|_{q_1} = l|_{q_2} = x$. Then $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2)$. Furthermore, $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m.q')$ for all non-junk one-step descendants q' of $u_m|_{p_m.q_1}$.*

Proof. We prove this by induction on the number of U-terms strictly below $u_m|_{p_m}$.

If there are no such U-terms our result holds by the definition of $\mathsf{t}_{\mathsf{opt}}$ and Lemma 20.

Otherwise there is one non-junk U-term $u_m|_{p_m.q_1.q}$. Let Q be all non-junk one-step descendants of $u_m|_{p_m.q_1.q}$ and therefore also of $u_m|_{p_m.q_2.q}$. For all non-junk terms $u_m|_{p_m.q_1.q.i}$, $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q.i) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2.q.i)$, and for all $q' \in Q$ $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q.i) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m.q'.q.i)$ by the inductive hypothesis.

We distinguish the following cases: If $u_m|_{p_m.q_1.q}$ is erased (w.r.t. junk terms), then also $u_m|_{p_m.q_2.q}$ is erased. Therefore, $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2.q)$. In this case either $Q = \emptyset$, or $u_{m+1}|_{p_m.q'.q}$ is also erased for all $q' \in Q$, hence $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m.q'.q)$.

If $u_m|_{p_m.q_1.q}$ is fully eliminated (w.r.t. junk terms), then so is $u_m|_{p_m.q_2.q}$. Therefore, also $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2.q)$. In this case $u_{m+1}|_{p_m.q'.q}$ is also fully eliminated for all $q' \in Q$, therefore $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m.q'.q)$.[3]

Therefore $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1.q) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2.q) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q'.q)$ for all non-junk U-terms $u_m|_{p_m.q_1.q}$ and its non-junk one-step descendants $q' \in Q$ and therefore also $\mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_1) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m.q_2) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m.q')$. □

This previous lemma implies that for unconditional rules $l\sigma \to_{\mathbb{U}_{\mathsf{seq}}(\mathcal{R})} r\sigma$ in a derivation we can translate the matcher σ into a matcher σ_{opt} using $\mathsf{t}_{\mathsf{opt}}$ so that $l\sigma_{\mathsf{opt}} \to_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} r\sigma_{\mathsf{opt}}$ even if l or r are non-linear.

Hence, we can translate rewrite steps from $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into rewrite steps in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$:

Lemma 22 (technical key result)**.** *Let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all non-junk U-terms are either erased or fully eliminated w.r.t. junk terms. If $u_m|_{p_m}$ is not a junk term then $\mathsf{t}_{\mathsf{opt}}(A, m, p_m) \to^* \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m)$.*

Proof. We use a case distinction on the rule $l \to r \in \mathbb{U}(\mathcal{R})$ that is applied in the mth rewrite step $u_m|_{p_m} \to_{p_m} u_{m+1}|_{p_m}$, i.e., $u_m|_{p_m} = l\sigma$ and $u_{m+1}|_{p_m} = r\sigma$.

If $l \to r$ is a (not necessarily linear) original rule, then by Lemma 21 $\mathsf{t}_{\mathsf{opt}}(A, m, p_m) = l\sigma_{\mathsf{opt}}$ and $\mathsf{t}_{\mathsf{opt}}(A, m+1, p_m) = r\sigma_{\mathsf{opt}}$. Since $l \to r \in \mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ also $l\sigma_{\mathsf{opt}} \to_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} r\sigma_{\mathsf{opt}}$.

If $l \to r$ is an introduction rule of the conditional rule $\alpha : l_\alpha \to r_\alpha \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$, then $r\sigma = U_j^{\mathsf{seq},\alpha}(s_1\sigma, \overrightarrow{X_1^{\mathsf{seq}}}\sigma)$ and $l\sigma = l_\alpha\sigma$.

[3] Observe that if $u_{m|_{p_m.q_1.q}}$ is eliminated, one of its non-junk one-step descendants might be erased while the other one is eliminated.

If the introduced U-term is erased, then (by the definition of $\mathsf{t}_{\mathsf{opt}}$) $\mathsf{t}_{\mathsf{opt}}(A, m+1, p_m) = \mathsf{t}_{\mathsf{opt}}(A, m, p_m) = l\sigma_{\mathsf{opt}}$ (by Lemma 21).

Otherwise the introduced U-term is fully eliminated. In this case $\mathsf{t}_{\mathsf{opt}}(A, m+1, p_m) = U_j^{\mathsf{opt},\alpha}(s_1\sigma_{\mathsf{opt}}, \overrightarrow{X_1^{\mathsf{seq}}}\sigma_{\mathsf{opt}})$ and $u_m|_{p_m} = l\sigma_{\mathsf{opt}}$. We therefore can apply the introduction rule of α in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.

The case of switch rules is dual to the case of introduction rules.

Finally, if $l\sigma \to r\sigma$ is an elimination step, then $u_m|_{p_m}$ is a fully eliminated U-term and $\mathsf{t}_{\mathsf{opt}}(A, m, p_m) = U_k^{\mathsf{opt},\alpha}(t_k\sigma_{\mathsf{opt}}, \overrightarrow{X_k^{\mathsf{opt}}}\sigma_{\mathsf{opt}})$. We can apply the corresponding elimination rule in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ of α: $U_k^{\mathsf{opt},\alpha}(t_k\sigma_{\mathsf{opt}}, \overrightarrow{X_k^{\mathsf{opt}}}\sigma_{\mathsf{opt}}) \to r_\alpha\sigma_{\mathsf{opt}}$ where $r_\alpha\sigma_{\mathsf{opt}} = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m)$. □

Next we prove a monotony property of $\mathsf{t}_{\mathsf{opt}}$:

Lemma 23. *Let $A : u_1 \to_{p_1} u_2 \cdots \to_{p_n} u_{n+1}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all non-junk U-terms are either erased or fully eliminated w.r.t. junk terms. Then $\mathsf{t}_{\mathsf{opt}}(A, m, \epsilon) \Vdash \mathsf{t}_{\mathsf{opt}}(A, m+1, \epsilon)$.*

Proof. Let $q \leq p_m$. We prove $\mathsf{t}_{\mathsf{opt}}(A, m, q) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q)$ by induction on $|p_m| - |q|$.

The base case $p_m = q$ holds by Lemma 22.

Otherwise, by the inductive hypothesis $\mathsf{t}_{\mathsf{opt}}(A, m, q.i) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q.i)$ for $q.i \leq p_m$ where $i \in \mathbb{N}$.

We distinguish two cases based on the root symbol of $u_m|_q$.

If $\mathsf{root}(u_m|_q) = f$ and $f \in \mathcal{F}$ is an original symbol, then also $\mathsf{root}(\mathsf{t}_{\mathsf{opt}}(A, m, q))$ is an original symbol. By the definition of $\mathsf{t}_{\mathsf{opt}}$,

$$\begin{aligned} \mathsf{t}_{\mathsf{opt}}(A, m, q) &= f(\mathsf{t}_{\mathsf{opt}}(A, m, q.1), \ldots, \mathsf{t}_{\mathsf{opt}}(A, m, q.\mathsf{ar}(f))), \text{and} \\ \mathsf{t}_{\mathsf{opt}}(A, m+1, q) &= \mathsf{t}_{\mathsf{opt}}(A, m, q)[\mathsf{t}_{\mathsf{opt}}(A, m+1, q.i)]_i \end{aligned}$$

Since $\mathsf{t}_{\mathsf{opt}}(A, m, q.i) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q.i)$ also $\mathsf{t}_{\mathsf{opt}}(A, m, q) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q)$.

Otherwise, $\mathsf{root}(u_m|_q)$ is a U-symbol, hence $u_m|_q = U_j^{\mathsf{seq},\alpha}(v, \overrightarrow{X_j^{\mathsf{seq}}}\sigma)$ is a U-term.

We distinguish two cases based on i. First, if $i = 1$ is inside the conditional argument, then the U-term is fully eliminated because otherwise $u_m|_{q.i}$ would be a junk term. Let $v = \mathsf{t}_{\mathsf{opt}}(A, m, q.1)$ and $v' = \mathsf{t}_{\mathsf{opt}}(A, m+1, q.1)$. By the inductive hypothesis $v \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} v'$. Now $\mathsf{t}_{\mathsf{opt}}(A, m, q) = U_j^{\mathsf{opt},\alpha}(v, \overrightarrow{X_j^{\mathsf{opt}}}\sigma_{\mathsf{opt}})$ and $\mathsf{t}_{\mathsf{opt}}(A, m+1, q) = U_j^{\mathsf{opt},\alpha}(v', \overrightarrow{X_j^{\mathsf{opt}}}\sigma_{\mathsf{opt}})$, therefore $\mathsf{t}_{\mathsf{opt}}(A, m, q) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q)$.

Otherwise $i > 1$. Let $x = \overrightarrow{X_j^{\mathsf{seq}}}[i-1]$ be the variable that is modified in this rewrite step.

There are two possible cases:

If $u_m|_q$ is fully eliminated then $\mathsf{t}_{\mathsf{opt}}(A, m, q) = U_j^{\mathsf{opt},\alpha}(v, \overrightarrow{X_j^{\mathsf{opt}}}\sigma_{\mathsf{opt}})$. Then $\mathsf{t}_{\mathsf{opt}}(A, m+1, q) = U_j^{\mathsf{opt},\alpha}(v, \overrightarrow{X_j^{\mathsf{opt}}}\tau\mathsf{opt})$ where $x\sigma_{\mathsf{opt}} \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} x\tau\mathsf{opt}$ by the inductive hypothesis, and $y\sigma_{\mathsf{opt}} = y\tau_{\mathsf{opt}}$ for all $y \neq x$. Hence, also in this case $\mathsf{t}_{\mathsf{opt}}(A, m, q) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q)$.

Finally, if $u_m|_q$ is erased (w.r.t. junk terms) let l be the left-hand side of the conditional rule α. Then, $\mathsf{t}_{\mathsf{opt}}(A, m, q) = l\sigma_{\mathsf{opt}}$ and $\mathsf{t}_{\mathsf{opt}}(A, m+1, q) = l\tau_{\mathsf{opt}}$. We apply the rewrite steps

$x\sigma_{\mathsf{opt}} \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} x\tau\mathsf{opt}$ to all occurrences of x in $l\sigma_{\mathsf{opt}}$ and therefore obtain and $y\sigma_{\mathsf{opt}} = y\tau_{\mathsf{opt}}$ $\mathsf{t}_{\mathsf{opt}}(A, m, q) \Vdash_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, q)$. □

Finally, we can translate complete derivations:

Lemma 24 (translations using $\mathsf{t}_{\mathsf{opt}}$)**.** *Let $A : u \to^* v$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all U-terms are either fully eliminated w.r.t. junk terms or erased w.r.t. junk terms. Then, $A_{\mathsf{opt}} : \mathsf{t}_{\mathsf{opt}}(A, 1, \epsilon) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, |A|+1, \epsilon)$ is a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$.*

Proof. By Lemma 23 $\mathsf{t}_{\mathsf{opt}}(A, m, \epsilon) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, m+1, \epsilon)$ for all $m \in \{1, \ldots, n\}$. □

One interesting class of derivations are derivations in which all U-terms have at most one non-junk one-step descendant (i.e. U-right-linear derivations (w.r.t. junk terms)). All U-terms are either erased or fully eliminated and therefore can be translated into derivations in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ using $\mathsf{t}_{\mathsf{opt}}$.

Lemma 25. *Let $A : u \to^* v$, $u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all U-terms have at most one non-junk one-step descendant. Then, all U-terms are either erased or fully eliminated w.r.t. junk terms.*

Proof. If A would contain a U-term that is neither erased nor fully eliminated, then there must be a U-term with two non-junk one-step descendants one of which is erased and the other one is eliminated. Yet, this contradicts our original assumption. □

Lemma 26. *Let $A : u_1 \to_{p_1} u_2 \to_{p_2} \cdots \to u_{n+1}$ be a derivation in some $\mathbb{U}_{\mathsf{seq}}\mathcal{R}$ such that $u_1 \in \mathcal{T}$ and all U-terms have at most one non-junk one-step descendant. Then in the derivation $A_{\mathsf{opt}} : \mathsf{t}_{\mathsf{opt}}(A, 1, \epsilon) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \mathsf{t}_{\mathsf{opt}}(A, 2, \epsilon) \to^*_{\mathbb{U}_{\mathsf{opt}}(\mathcal{R})} \cdots \mathsf{t}_{\mathsf{opt}}(A, n+1, \epsilon)$ all U-terms have at most one one-step descendant.*

Proof. By Lemma 25 we can translate A into a derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ using $\mathsf{t}_{\mathsf{opt}}$.

Consider the rewrite step $u_i \to_{p_i} u_{i+1}$ and let $u_m|_p$ be a non-junk U-term that is eliminated.

$\mathsf{t}_{\mathsf{opt}}(A, m, p)$ returns a term that might occur in multiple positions in $\mathsf{t}_{\mathsf{opt}}(A, m, \epsilon)$ because erased U-terms are translated into the possibly non-linear left-hand side of the corresponding conditional rule. We show that all occurrences of $\mathsf{t}_{\mathsf{opt}}(A, m, p)$ have at most one one-step descendant in the translated derivation.

We distinguish the following cases based on the rule applied in A in the rewrite step $u_m \to_{p_m} u_{m+1}$. First, if $p \parallel p_m$ or $p \leq p_m$ then our result holds.

Otherwise, $p_m < p$. We distinguish the following cases based on the rule applied.

If the applied rule is a switch rule or an introduction of an erased (w.r.t. junk terms) U-term then $\mathsf{t}_{\mathsf{opt}}(A, m, p_m) = \mathsf{t}_{\mathsf{opt}}(A, m+1, p_m)$. In this case we repeat our argument for $u_{m+1}|_{p'}$ where p' is the unique non-junk one-step descendant of $u_m|_p$.

Finally, if it is a an elimination rule, a switch rule or introduction rule applied to an eliminated (w.r.t. junk terms) U-term or an original rule then the corresponding rule of $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ is applied in A_{opt}. Since p has at most one non-junk descendant in A, all its corresponding positions in A_{opt} therefore also have at most one descendant. □

We now can prove soundness for derivations in $\mathbb{U}_{\mathsf{seq}}$ using our soundness results for $\mathbb{U}_{\mathsf{opt}}$:

Lemma 27 (soundness for U-right-linear derivations)**.** *Let $\mathcal{R}$ be a sorted DCTRS and $A : u \to^*_{\mathbb{U}_{\mathsf{seq}}\mathcal{R}} v$ be a derivation such that $u \in \mathcal{T}$ and all U-terms have at most one non-junk one-step descendant. Then $u \to^*_{\mathcal{R}} v$.*

Proof. By Lemma 24 and Lemma 20 we obtain a derivation $A_{\mathsf{t}_{\mathsf{opt}}} : u \to^*_{\mathbb{U}_{\mathsf{opt}}\mathcal{R}} \mathsf{tb}(v)$ such that all U-terms are eliminated. By Lemma 26 all U-terms have at most one one-step descendant. Since $\mathcal{R}$ is sorted we finally can apply Lemma 19 and obtain $u \to^*_{\mathcal{R}} v$. □

4.7.5 Weakly Right-Linear DCTRSs

$\mathbb{U}_{\mathsf{seq}}$ is sound for $\mathbb{U}_{\mathsf{seq}}$-right-linear DCTRSs. This is a consequence of Theorem 2 because every $\mathbb{U}_{\mathsf{seq}}$-right-linear DCTRS is also left-separate. Left-separateness is a very restrictive property and $\mathbb{U}_{\mathsf{seq}}$-right-linearity is even more restrictive. Therefore, this result is of less practical relevance. Yet we can prove soundness for an weaker right-linearity-property.

In [7] we introduced the notion of weak left-linearity to allow non-left-linear rules to a certain extend. We can define a dual property for right-linearity, weak right-linearity.

The symmetric equivalent of weak left-linearity allows non-right-linear switch and elimination rules, provided that the extra variables have not been used before. This is not possible for deterministic rules. By restricting such systems to deterministic CTRSs we obtain the following definition:

Definition 19 (weak right-linearity)**.** *A conditional deterministic rule $\alpha : l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ is weakly right-linear, if for all $x \in \mathcal{V}ar(\alpha)$, $|s_1, \ldots, s_k, r|_x \leq 1$.*

This definition allows non-right-linear switch rules if the non-linear variable is not used in any other conditional argument or the right-hand side of the conditional rule. In $\mathbb{U}_{\mathsf{opt}}$, this implies that the variable would not be encoded anymore provided that the DCTRS is sorted. Therefore, $\mathbb{U}_{\mathsf{opt}}$-right-linearity is equivalent to weak right-linearity for sorted DCTRSs.

Example 36. *The following CTRS resembles the "bubble-sort"-Example of [32] and defines a sorted list data structure in which multiple occurrences of the same argument are removed.*[4]

$$\mathcal{R} = \left\{ \begin{array}{c} x :: y :: rest \to z_1 :: z_2 :: rest \Leftarrow orient(x, y) \to^* \langle z_1, z_2 \rangle \\ x :: x :: rest \to x :: rest \\ orient(s(x), s(y)) \to \langle s(z_1), s(z_2) \rangle \Leftarrow orient(x, y) \to^* \langle z_1, z_2 \rangle \\ orient(s(x), 0) \to \langle 0, s(x) \rangle \end{array} \right\}$$

[4] Observe that the CTRS in [32] is not weakly right-linear. We therefore replace the comparison by the new partial $orient$-function that switches position of two elements if the first element is greater than the second.

$\mathcal{R}$ is weakly right-linear and also sorted. The transformed CTRS using $\mathbb{U}_{\mathsf{seq}}$ is the following:

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{rcl} x :: y :: rest & \to & U_1^{\mathsf{seq},\alpha}(orient(x,y),x,y,rest) \\ U_1^{\mathsf{seq},\alpha}(\langle z_1,z_2\rangle,x,y,rest) & \to & z_1 :: z_2 :: rest \\ x :: x :: rest & \to & x :: rest \\ orient(s(x),s(y)) & \to & U_1^{\mathsf{seq},\beta}(orient(x,y),x,y) \\ U_1^{\mathsf{seq},\beta}(\langle z_1,z_2\rangle,x,y) & \to & \langle s(z_1),s(z_2)\rangle \\ orient(s(x),0) & \to & \langle 0,s(x)\rangle \end{array} \right\}$$

Observe that $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is not right-linear. The transformed CTRS using $\mathbb{U}_{\mathsf{opt}}$ is:

$$\mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{rcl} x :: y :: rest & \to & U_1^{\mathsf{opt},\alpha}(orient(x,y),rest) \\ U_1^{\mathsf{opt},\alpha}(\langle z_1,z_2\rangle,rest) & \to & z_1 :: z_2 :: \langle rest\rangle \\ x :: x :: rest & \to & x :: rest \\ orient(s(x),s(y)) & \to & U_1^{\mathsf{opt},\beta}(orient(x,y)) \\ U_1^{\mathsf{opt},\beta}(\langle z_1,z_2\rangle) & \to & \langle s(z_1),s(z_2)\rangle \\ orient(s(x),0) & \to & \langle 0,s(x)\rangle \end{array} \right\}$$

Weak right-linearity does not imply sortedness for deterministic CTRSs, and hence we obtain unsoundness for non-sorted weakly right-linear DCTRSs.

Example 37. *Consider the following weakly right-linear CTRS that resembles the CTRS of Example 18:*

$$\mathcal{R} = \left\{ \begin{array}{l} a \to b \\ s(a) \to t(b) \\ f(x) \to c \Leftarrow s(x) \to^* t(x) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{rcl} a & \to & b \\ s(a) & \to & t(b) \\ f(x) & \to & U_1^{\alpha}(s(x),x) \\ U_1^{\alpha}(t(x),x) & \to & c \end{array} \right\}$$

The unraveled TRS now gives rise to the following derivation that is unsound by the same argument as the derivation in Example 18:

$$f(a) \to U_1^{\alpha}(s(a),a) \Rrightarrow U_1^{\alpha}(t(b),b) \to c$$

Observe that the CTRS is not $\mathbb{U}_{\mathsf{opt}}$-right-linear:

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{rcl} a & \to & b \\ s(a) & \to & t(b) \\ f(x) & \to & U_1^{\alpha}(s(x),x) \\ U_1^{\alpha}(t(x),x) & \to & c \end{array} \right\}$$

In [25] it was shown that $\mathbb{U}_{\mathsf{opt}}$ is sound for $\mathbb{U}_{\mathsf{opt}}$-right-linear and $\mathbb{U}_{\mathsf{opt}}$-non-erasing DCTRSs. Lemma 19 implies soundness for derivations in some $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ provided all U-terms are eliminated. Since $\mathsf{t}_{\mathsf{opt}}$ replaces all erased U-terms by an original term we first prove that we can translate derivations in weak right-linear DCTRSs into derivations in $\mathbb{U}_{\mathsf{opt}}$ and then show that these derivations satisfy the assumptions of Lemma 19.

Weakly right-linear DCTRSs are usually not $\mathbb{U}_{\mathsf{seq}}$-right-linear. Therefore, terms might have multiple one-step descendants. Yet, if a U-term has more than one one-step descendant, then at least one of them is a junk term:

Lemma 28. *Let $A : u \to^* v$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly right-linear, sorted DCTRS. Then, every U-term in A has at most one non-junk one-step descendant.*

Proof. Let A be the derivation $u_1 \to_{p_1} u_2 \to_{p_2} \cdots \to_{p_n} u_n$ and let the mth rewrite step be $u_m \to_{p_m, l\to r} u_{m+1}$ such that there is a U-term $u_m|_{p_m.q}$ with more than one one-step descendant.

By the definition of weak right-linearity, $l \to r$ is either an introduction or switch rule, therefore $u_m|_{p_m} = r\sigma = U_j^{\mathsf{seq},\alpha}(s_j\sigma, \overrightarrow{X_j^{\mathsf{seq}}}\sigma)$.

Let $x \in \mathcal{V}ar(r)$ be the variable that contains the binding containing the U-term, i.e., $x = r|_{q'}$ and $q' \leq q$. Then $|s_j|_x = 1$ and $|s_{j+1}, \ldots, s_k, r|_x = 0$. Because of sortedness also $|t_j, \ldots, t_k|_x = 0$. Furthermore, $x \in X_j^{\mathsf{seq}}$, therefore $u_m|_{p_m.q}$ has two one-step descendants, one in the variable arguments and one in the conditional arguments.

We distinguish two cases: If the U-term at $u_m|_{p_m.q}$ is eliminated w.r.t. junk terms, then the variable argument containing x is a junk term since $|s_j, \ldots, s_k, t_{j+1}, \ldots, t_k, r|_x = 0$.

Otherwise, if the U-term is erased, then the conditional argument is a junk term. Therefore, only one of these one-step descendants is not a junk term. □

We now combine our results for $\mathbb{U}_{\mathsf{opt}}$ with our translation of derivations and obtain the following soundness result:

Lemma 29 (soundness of weakly right-linear, sorted DCTRSs)**.** *Let $A : u \to^* v$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly right-linear, sorted DCTRS. Then there is a derivation $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$.*

Proof. By Lemma 28 all U-terms in the derivation A have at most one non-junk one-step descendant. By Lemma 27 we therefore obtain $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$. □

Theorem 6 (soundness of weakly right-linear, sorted DCTRSs)**.** $\mathbb{U}_{\mathsf{seq}}$ *is sound w.r.t.* tb *for weakly right-linear, sorted DCTRSs.*

Proof. Straightforward from Lemma 29. □

We also obtain soundness w.r.t. joinability:

Theorem 7 (soundness w.r.t. joinability of weakly right-linear, sorted DCTRSs)**.** $\mathbb{U}_{\mathsf{seq}}$ *is sound w.r.t. joinability for weakly right-linear, sorted DCTRSs.*

Proof. Straightforward from Lemma 29 and Lemma 1. □

4.7.6 Weakly Left-Linear DCTRSs

In [7] we introduced the notion of *weak left-linearity* that allows non-left-linear rules provided all non-linear variables one the left-hand side of the rule are erased. Such systems are practically relevant because they can contain rules like $eq(x, x) \to true$. In [8] we extended the notion of weak left-linearity to DCTRSs:

Definition 20 (weak left-linearity [8, Definition 19])**.** *A conditional deterministic rule $\alpha : l \to r \Leftarrow s_1 \to^* t_1, \ldots, s_k \to^* t_k$ is weakly left-linear, if $|l, t_1, \ldots, t_k|_x > 1 \Rightarrow |s_1, \ldots, s_k, r|_x = 0$ for all $x \in \mathcal{V}ar(\alpha)$.*

In contrast to weak right-linearity, weak left-linearity implies sortedness for deterministic CTRSs:

Lemma 30. *Let $\mathcal{R}$ be a weakly left-linear DCTRS. Then, $\mathcal{R}$ is sorted.*

Proof. Let $x \in \mathcal{V}ar(s_i)$ be a variable in the left-hand side of a condition. The definition of deterministic CTRSs implies that $x \in \mathcal{V}ar(l, t_1, \ldots, t_{i-1})$. Definition 20 then implies that $x \notin \mathcal{V}ar(t_i, \ldots, t_k)$. □

The class of weakly left-linear DCTRS is of high interest because every left-linear join 1-CTRS can be simulated using a weak left-linear DCTRS. Therefore, soundness for an unraveling of weakly left-linear DCTRSs implies soundness for (weakly) left-linear join 1-CTRSs.

In weakly left-linear DCTRSs, non-left-linear variables can be used to a certain extend, but the syntactic restriction ensures that their variable binding and all terms derived from them are always wrapped inside a U-term:

Example 38. *Consider the following weakly left-linear DCTRS and its unraveled TRS:*

$$\mathcal{R} = \left\{ \begin{array}{rl} up(x) & \to x \\ down(x) & \to x \\ up(x) & \to up(s(x)) \\ down(s(x)) & \to down(x) \\ between(x, y, z) & \to true \Leftarrow up(x) \to^* y, down(z) \to^* y \end{array} \right\}$$

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{rl} up(x) & \to x \\ down(x) & \to x \\ up(x) & \to up(s(x)) \\ down(s(x)) & \to down(x) \\ between(x, y, z) & \to U_1^\alpha(up(x), x, y, z) \\ U_1^\alpha(y, x, y, z) & \to U_2^\alpha(down(z), x, y, z) \\ U_2^\alpha(y, x, y, z) & \to true \end{array} \right\}$$

Observe that although $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is not weakly left-linear, the binding of the non-linear variable y cannot be propagated to any term outside of the U-term.

In order to prove soundness of a weakly left-linear DCTRS $\mathcal{R}$ we translate derivations in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ into derivations of which we know that they are sound. In order to apply our translation $\mathsf{t}_{\mathsf{opt}}$, we need to ensure that all U-terms are fully eliminated or erased. Yet, even for left-linear systems we cannot ensure this property in general:

Example 39. *Consider the left-linear DCTRS $\mathcal{R}$ and its unraveled TRS*

$$\mathcal{R} = \left\{ \begin{array}{c} f(x) \to x \Leftarrow x \to^* a \\ g(x) \to h(x,x) \\ h(x,y) \to i(x) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{c} f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(a,x) \to x \\ g(x) \to h(x,x) \\ h(x,y) \to i(x) \end{array} \right\}$$

Consider the following derivation:

$$g(f(a)) \to g(\overbrace{U_1^\alpha(a,a)}^{(1)}) \to h(\overbrace{U_1^\alpha(a,a)}^{(2)}, \overbrace{U_1^\alpha(a,a)}^{(3)}) \to h(a, \overbrace{U_1^\alpha(a,a)}^{(4)}) \to i(a)$$

The U-term at (2) is eliminated while the U-term at (3) is erased. Therefore, the term at (1) is both eliminated and erased.

We therefore use a different proof attempt than for weakly right-linear CTRSs. We show that we can transform derivations so that all U-terms have at most one one-step descendants. Since this implies that the derivation has at most one non-junk one-step descendant we then can prove soundness using Lemma 27. We therefore do not rely on the notion of junk terms for weakly left-linear CTRSs.

Observe that weak left-linearity of a DCTRS does not imply weak left-linearity of its unraveled TRS because switch rules and introduction rules also contain non-left-linear variables on their left-hand sides. These variables are preserved on the right-hand sides of such rules. Since these variables play an important role in our soundness proof we will refer to a variable argument x in a U-term $U_j^\alpha(\ldots,x,\ldots)$ as *non-left-linear variable argument* if there is an introduction or switch rule $l \to U_i^\alpha(\ldots,x,\ldots)$ such that $i \leq j$ and $|l|_x > 1$. All other variable arguments are *left-linear variable arguments*. Observe that a variable argument can be a left-linear variable argument and become a non-left-linear variable argument in some later switch rule.

The following observations are essential for our proof and directly derive from the definition of weak left-linearity and non-left-linear variable arguments:

Observation 1. *All descendants of terms inside a non-left-linear variable argument are also inside a non-left-linear variable argument.*

Observation 2. *All non-left-linear variable arguments x are linear on the right-hand side of introduction and switch rules.*

In the following we will prove that we can convert rewrite sequences in weakly left-linear CTRSs into rewrite sequences in which all U-terms have at most one one-step descendant.

Let in the following v be a U-term and u be its introduction term. Consider the derivation $l[u] \to l[v] \to r[v,v]$. The U-term v has two one-step descendants. We can move the introduction step past the non-right-linear rule, thus avoiding that v is duplicated: $l[u] \to r[u,u] \Rsh r[v,v]$. In detail, we can shift the introduction steps behind so that the introduction term is duplicated instead of the U-term.

Yet, for this shift-behind, every U-term must have a unique introduction term. Since we allow non-left-linear introduction and switch rules there may be multiple introduction terms:

$$s[l[u_1], l[u_2]] \to^* s[l[v], l[v]] \to t[l[v]] \to t[r[v,v]]$$

Observe that in this case t is a U-rooted term and $l[v]$ is inside a non-left-linear variable argument in t.

In the second part of the derivation $s[l[v], l[v]] \to t[l[v]] \to t[r[v,v]]$ we can delay the switch or introduction step $s[l[v], l[v]] \to t[l[v]]$ and obtain

$$s[l[v], l[v]] \Rsh s[r[v,v], r[v,v]] \to t[r[v,v]]$$

Now, the introduction terms are unique for all U-terms and we can shift behind the introduction step twice:

$$s[l[u_1], l[u_2]] \to^* s[r[u_1,u_1], r[u_2,u_2]] \Rsh s[r[v,v], r[v,v]] \to t[r[v,v]]$$

Using these shifts of rewrite steps we can transform all derivations in weakly left-linear DCTRSs into a derivation in which U-terms have at most one one-step descendant. In the following example we illustrate how to apply these shift steps.

Example 40. *Consider the following DCTRS and its unraveled TRS:*

$$\mathcal{R} = \left\{ \begin{array}{c} a \to c \to e \\ \searrow \quad \swarrow \\ b \to d \to k \\ f(x) \to x \Leftarrow x \to^* e \\ g(x,x) \to C \Leftarrow A \to^* B \\ h(x) \to i(x,x) \end{array} \right\} \qquad \mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{c} a \to c \to e \\ \searrow \quad \swarrow \\ b \to d \to k \\ f(x) \to U_1^\alpha(x,x) \\ U_1^\alpha(e,x) \to x \\ g(x,x) \to U_1^\beta(A,x) \\ U_1^\beta(B,x) \to C \\ h(x) \to i(x,x) \end{array} \right\}$$

Observe, that $\mathcal{R}$ is weakly left-linear while $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is not. $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ gives rise to the following derivation:

$$\begin{aligned} A : g(h(f(a)), h(f(b))) &\Rsh g(h(U_1^\alpha(a,a)), h(U_1^\alpha(b,b))) \to^* g(h(U_1^\alpha(c,d)), h(U_1^\alpha(c,d))) \\ &\to U_1^\beta(A, h(U_1^\alpha(c,d))) \to U_1^\beta(A, i(U_1^\alpha(c,d), U_1^\alpha(c,d))) \\ &\to^* U_1^\beta(A, i(U_1^\alpha(e,d), U_1^\alpha(k,k))) \to U_1^\beta(A, i(d, U_1^\alpha(k,k))) \end{aligned}$$

We want to show that there is an equivalent derivation such that all U-terms have at most one one-step descendant. In A, the inner U-term in the term $U_1^\beta(A, h(U_1^\alpha(c,d)))$ has two one-step descendants. We cannot shift behind the introduction step of the inner U-term because it has two introduction terms in A. We first need to delay the application of the non-left-linear introduction rule and thereby prevent rewrite steps inside non-left-linear variable arguments. We therefore apply all rewrite steps inside the non-left-linear variable argument before the introduction step (in all parallel positions) and obtain

$$\begin{aligned} g(h(U_1^\alpha(c,d)), h(U_1^\alpha(c,d))) &\to^* g(i(U_1^\alpha(c,d), U_1^\alpha(c,d)), i(U_1^\alpha(c,d), U_1^\alpha(c,d))) \\ &\to^* g(i(d, U_1^\alpha(k,k)), i(d, U_1^\alpha(k,k))) \end{aligned}$$

Only then we apply the introduction rule to the g-rooted term:

$$g(i(d, U_1^\alpha(k,k)), i(d, U_1^\alpha(k,k))) \to^* U_1^\beta(A, i(d, U_1^\alpha(k,k)))$$

Figure 4.6 illustrates the transformation of the derivation along with the successor relation of all U-terms.

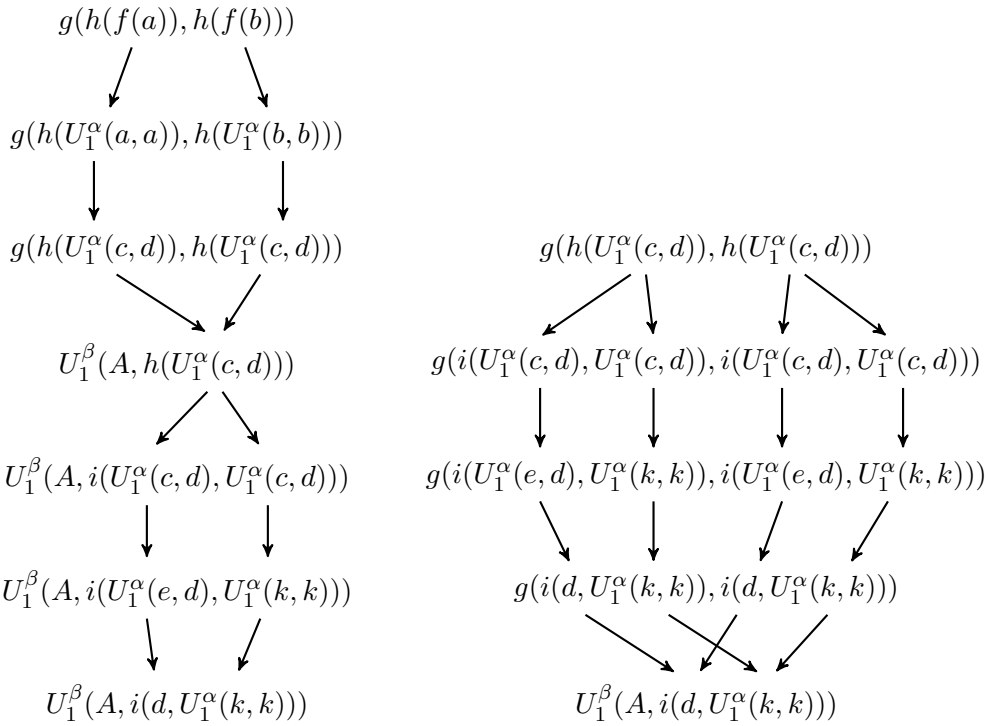

Figure 4.6: Delaying the introduction step in Example 40

Now, we eliminated all rewrite steps inside non-left-linear variable arguments. Therefore, all U-terms have one unique introduction term and we can shift the corresponding introduction

steps behind the application of the non-right-linear rule $h(x) \to i(x,x)$*:*

$$\begin{aligned} g(h(f(a)),h(f(b))) &\to^* g(i(f(a),f(a)),i(f(b),f(b))) \\ &\to^* g(i(U_1^\alpha(a,a),U_1^\alpha(a,a)),i(U_1^\alpha(b,b),U_1^\alpha(b,b))) \end{aligned}$$

This step is illustrated in Figure 4.7.

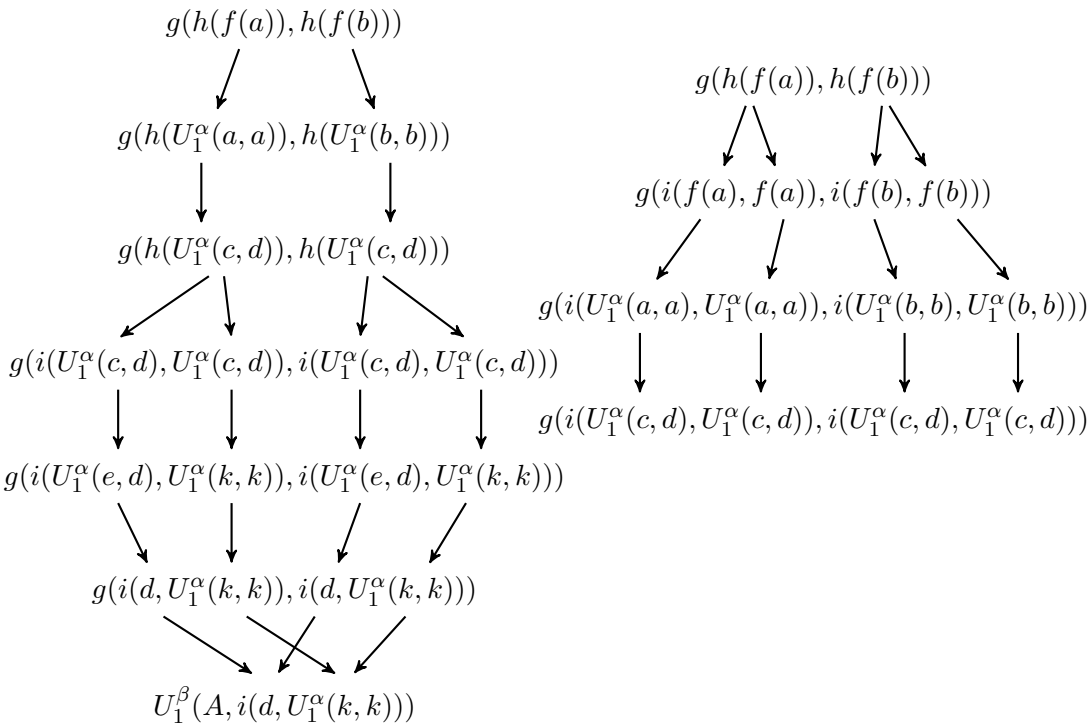

Figure 4.7: Shifting behind the introduction steps in Example 40

In total we obtain the derivation

$$\begin{aligned} g(h(f(a)),h(f(b))) &\to^* g(i(f(a),f(a)),i(f(b),f(b))) \\ &\to^* g(i(U_1^\alpha(a,a),U_1^\alpha(a,a)),i(U_1^\alpha(b,b),U_1^\alpha(b,b))) \\ &\to^* g(i(U_1^\alpha(c,d),U_1^\alpha(c,d)),i(U_1^\alpha(c,d),U_1^\alpha(c,d))) \\ &\to^* g(i(U_1^\alpha(e,d),U_1^\alpha(k,k)),i(U_1^\alpha(e,d),U_1^\alpha(k,k))) \\ &\to^* g(i(d,U_1^\alpha(k,k)),i(d,U_1^\alpha(k,k))) \to U_1^\beta(A,i(d,U_1^\alpha(k,k))) \end{aligned}$$

Now, all U-terms have at most one one-step descendant.

In order to prove that we can remove all rewrite steps in non-left-linear variable arguments we extract derivations in U-terms from other derivations and then rebuild these extracted derivations. We first show that we can group rewrite steps in U-terms in derivations without rewrite steps in non-left-linear variable arguments. This corresponds to one shift-behind step.

Lemma 31. *Let $A : u \to^* v, u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly-left-linear DCTRS and there are no rewrite steps in non-left-linear variable arguments in A. Let $v|_p$ be a U-term that is not inside a non-left-linear variable argument. Then, there is a derivation $A' : u \to^* w \to_q v$ in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that $p \leq q$, and such that A' does not contain rewrite steps in non-left-linear variable arguments.*

Proof. Let A be the derivation $u_1 \to_{p_1} \cdots \to_{p_n} u_{n+1}$ where $u_{n+1}|_p$ is a U-term. Let m be the largest m such that $q \leq p_m$ where $u_{m+1}|_q$ is an ancestor of $u_{n+1}|_p$.[5] Observe that m and q are unambiguous in A because $u_{n+1}|_p$ is not inside a non-left-linear variable argument.

We prove by induction on $n - m$ that we can shift the rewrite step $u_m \to u_{m+1}$ to the end of the derivation A.

If $n = m$, then $p = q$, $p_m = p_n$ and therefore $p \leq p_n$.

Otherwise, we show by case distinction on q and p_{m+1} that we can switch the mth and $m + 1$st rewrite step. If $q \parallel p_{m+1}$, then $u_m = C[u_m|_q]_q[u_m|_{p_{m+1}}]_{p_{m+1}}$ and

$$u_{m+2} = C[u_{m+1}|_q]_q[u_{m+2}|_{p_{m+1}}]_{p_{m+1}}.$$

Let $u'_{m+1} = C[u_m|_q]_q[u_{m+2}|_{p_{m+1}}]_{p_{m+1}}$. We obtain the derivation $u_m \to_{p_{m+1}} u'_{m+1} \to_{p_m} u_{m+2}$ and in total

$$u_1 \to^* u_m \to_{p_{m+1}} u'_{m+1} \to_{p_m} u_{m+2} \to^* u_{n+1}$$

Now, the last rewrite step in which the U-term is rewritten is $u'_{m+1} \to_{p_m} u_{m+2}$. Since the length of $u'_{m+1} \to^* u_{n+1}$ is $n - (m + 1)$ we can apply the inductive hypothesis. The case $q \leq p_{m+1}$ is not possible by assumption.

Finally, if $p_{m+1} < q$, then $u_m = u_{m+1}[u_m|_q]_q$, $u_{m+1} = u_{m+1}[u_{m+1}|_q]_q$ and $u_{m+2} = u_{m+2}[u_{m+1}|_q]_Q$ where Q are all one-step descendants of q.

First we show that no $q' \in Q$ is inside a non-left-linear variable argument. Observe that all rules introducing non-left-linear variable arguments (introduction and switch rules) are right-linear for these variables by the definition of weak left-linearity and that $u_m|_q$ is not inside a non-left-linear variable argument. Therefore, also $u_{m+1}|_q$ is not inside a non-left-linear variable argument. Hence, if there is one one-step descendant of $u_{m+1}|_q$ inside a non-left-linear variable argument, then this term is the only one-step descendant of $u_{m+1}|_q$. Yet, this one-step descendant also must be an ancestor of $u_{n+1}|_p$ which contradicts the assumption that $u_{n+1}|_p$ is not inside a non-left-linear variable argument. Hence, no one-step descendant of $u_{m+1}|_q$ is inside a non-left-linear variable argument.

Let $Q = \{q_1, \ldots, q_k\}$ such that q_k is the (unique) ancestor of $u_{n+1}|_p$ in u_{m+2}. Then we can apply the $m - 1$st rewrite step already to u_m:

$$u_{m+1}[u_m|_q]_q \to_{p_{m+1}} u_{m+2}[u_m|_q]_Q$$

Let $p_m = q.q'$, then we apply the rewrite step $u_m|_q \to u_{m+1}|_q$ to all one-step-descendants of the U-term:

$$\begin{aligned} u_{m+2}[u_m|_q]_{\{q_1,\ldots,q_k\}} &\to_{q_1.q'} u_{m+2}[u_{m+1}|_q]_{q_1}[u_m|_q]_{\{q_2,\ldots,q_k\}} \\ &\to_{q_2.q'} u_{m+2}[u_{m+1}|_q]_{\{q_1,q_2\}}[u_m|_q]_{\{q_3,\ldots,q_k\}} \cdots \\ &\to_{q_{k-1}.q'} u_{m+2}[u_{m+1}|_q]_{\{q_1,\ldots,q_{k-1}\}}[u_m|_q]_{q_k} \to_{q_k.q'} u_{m+2}[u_{m+1}|_q]_Q \end{aligned}$$

[5] By the common definition of ancestors, $u_m|_q$ is not an ancestor of $u_{n+1}|_p$ if $q = p_m$.

We therefore obtain in total the following derivation:

$$u_1 \to^* u_{m+2}[u_{m+1}|_q]_{\{q_1,\ldots,q_{k-1}\}}[u_m|_q]_{q_k} \to_{q_k.q'} u_{m+2} \to^* u_{n+1}$$

Now, the last rewrite step in which the U-term is rewritten is

$$u_{m+2}[u_{m+1}|_q]_{\{q_1,\ldots,q_{k-1}\}}[u_m|_q]_{q_k} \to_{q_k.q'} u_{m+2}.$$

Since the length of $u_{m+2}[u_{m+1}|_q]_{\{q_1,\ldots,q_{k-1}\}}[u_m|_q]_{q_k} \to^* u_{n+1}$ is $n-(m+1)$ we can apply the inductive hypothesis.

Since no $q_i.q'$ ($i \in \{1,\ldots,k\}$) is inside a non-left-linear variable argument the final derivation also does not contain any rewrite step in a non-left-linear variable argument. □

Next, we use this result to isolate derivations in U-terms from the rest of the derivation.

Lemma 32. *Let $A : u \to^* v, u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly-left-linear DCTRS and there are no rewrite steps in non-left-linear variable arguments in A. Let $v|_p$ be a U-term that is not inside a non-left-linear variable argument. Then, there are derivations $A' : u \to^* v[l\sigma]_p$ and $B : l\sigma \to^* v|_p$ in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that A' and B do not contain rewrite steps in non-left-linear variable arguments where $l\sigma$ is the (unique) introduction term of $v|_p$.*

Proof. Let A be the derivation $u_1 \to_{p_1} \cdots \to_{p_n} u_{n+1}$ where $u_{n+1}|_p$ is a U-term. We assume that $p \leq p_n$ because of Lemma 31.

Let $\mathsf{ti}(A, n+1, p) = \{(m+1, q)\}$ (since $u_{n+1}|_p$ is not inside a non-left-linear variable argument there is only one introduction term), i.e, $u_m|_q = l\sigma$ is the unique introduction term of $u_{n+1}|_p$.

By induction on $n-m$, if $n = m$, then $p_n = p$, A' is the derivation $u_1 \to^* u_n$ and B only contains the introduction step $l\sigma \to u_{n+1}|_p$.

Otherwise, $u_n|_p$ is also a U-term (since $p \leq p_n$). We therefore can apply the inductive hypothesis to $u_1 \to^* u_n$ and $u_n|_p$ and obtain derivations $u_1 \to^* u_n[l\sigma]_p$ and $l\sigma \to^* u_n|_p \to u_{n+1}|_p$. □

We can split derivations into two derivations and thereby isolate rewrite steps in U-terms. Next we show that we can move rewrite steps in non-left-linear variable arguments into these isolated derivations and thereby eliminate these rewrite steps. This corresponds to the delaying of switch and introduction steps that rise rise to non-left-linear variable arguments. Figure 4.6 illustrates multiple applications of this step.

Lemma 33. *Let $A : u \to^* v, u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly-left-linear DCTRS. Then, there is a derivation $A' : u \to^* v$ such that there are no rewrite steps in non-left-linear variable arguments.*

Proof. Let A be the derivation $u_1 \to_{p_1} \cdots \to_{p_n} u_{n+1}$. We prove this result by induction on the length n of A. If $n = 1$ there are no rewrite steps in non-left-linear variable arguments because u_1 does not contain any U-terms and therefore also no non-left-linear variable arguments.

Otherwise, by the inductive hypothesis, there are no rewrite steps in non-left-linear variable arguments in $u_1 \to^* u_n$. If p_n is not inside a non-left-linear variable argument we are done.

Otherwise, there is a U-term $u_n|_p$ such that $u_n|_{p.i}$ ($i \in \mathbb{N}$) is a non-left-linear variable argument and $p.i.p' = p_n$. In case of nested non-left-linear variable arguments let p be the innermost such U-term and let d be the number of nested non-left-linear variable arguments. We use an inductive argument on d in the following.

Let $l\sigma$ be the introduction term of $u_n|_p$. We can apply Lemma 32 to the derivation $u_1 \to^* u_n$ and the U-term $u_n|_p$ and obtain derivations $A' : u_1 \to^* u_n[l\sigma]_p$ and $l\sigma \to^* u_n|_p$ without rewrite steps in non-left-linear variable arguments.

Let in the following for better readability $B : v_1 \to^*_{q_1} \cdots v_k \to_{q_k} v_{k+1}$ be the derivation $l\sigma \to^* u_n|_p \to u_{n+1}|_p$ where $q_k = i.p'$. Observe that the only rewrite step inside a non-left-linear variable argument in B is the last rewrite step $v_k \to_{q_k} v_{k+1}$.

We prove that we can transform B into a derivation without rewrite steps in non-left-linear variable arguments by induction on the length k of B.

The base case is $B : v_1 \to_{l\to r} v_2 \to_{i.p'} v_3$ where $v_1 \to v_2$ is the introduction step that introduces the non-left-linear variable argument. Let $r|_i = x$ be the corresponding variable. Then $|l|_x > 1$ and $|r|_x = 1$. Therefore, $v_1 = l\sigma$, $v_2 = r\sigma$ and $v_3 = r\sigma'$ where $x\sigma' = x\sigma[v_3|_{i.p'}]_{p'}$ and $y\sigma = y\sigma'$. Since $x\sigma \to x\sigma'$ we obtain the following derivation:

$$l\sigma \Vdash l\sigma' \to r\sigma'$$

This derivation now does not contain rewrite steps in non-left-linear variable arguments.

For the induction step we use a case distinction on q_{k-1} and i where $q_k = i.p'$.

The case $i \leq q_{k-1}$ is not possible because $v_1 \to^* v_k$ does not contain rewrite steps in non-left-linear variable arguments.

If $i \parallel q_{k-1}$, we can simply swap the $k-1$st and the kth rewrite step and then apply the inductive hypothesis, dual to the corresponding case in the proof of Lemma 31.

The only remaining case is $q_{k-1} = \epsilon$. In this case the $k-1$st rewrite step in B is a switch step, i.e., $v_{k-1} = U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau)$, $v_k = U^\alpha_{j+1}(s_{j+1}\tau, \overrightarrow{X_{j+1}}\tau)$ and $v_{k+1} = U^\alpha_{j+1}(s_{j+1}\tau', \overrightarrow{X_{j+1}}\tau')$ where $x\tau' = x\tau[v_{k+1}|_{i.p'}]_{p'}$ and $y\tau = y\tau'$ for all $y \neq x$.

In this case, $x \notin \mathcal{V}ar(s_{j+1})$ by the definition of weak left-linearity.

We distinguish two cases. If $x \notin X_j$ we obtain the derivation

$$v_1 \to^* U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau) \to U^\alpha_j(t_j\tau', \overrightarrow{X_j}\tau) \to U^\alpha_{j+1}(s_{j+1}\tau', \overrightarrow{X_{j+1}}\tau')$$

which does not contain rewrite steps in non-left-linear variable arguments.

Otherwise, $x \in X_j$ and we obtain the derivation

$$v_1 \to^* U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau) \to U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau')$$

If x is a non-left-linear variable argument in X_j we can apply the inductive hypothesis and obtain a derivation $v_1 \to^* U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau')$ without rewrite steps in non-left-linear variable arguments.

Since $x\tau \to x\tau'$ does not contain rewrite steps in non-left-linear variable arguments (because of the assumption that $u_n|_{p.i}$ is the innermost such argument), also the following derivation does not contain such rewrite steps

$$U^\alpha_j(t_j\tau, \overrightarrow{X_j}\tau') \Vdash U^\alpha_j(t_j\tau', \overrightarrow{X_j}\tau') \to U^\alpha_{j+1}(s_{j+1}\tau', \overrightarrow{X_{j+1}}\tau')$$

We therefore obtain a derivation $B' : v_1 \to^* v_{k+1}$ without rewrite steps in non-left-linear variable arguments.

We conclude the proof by recombining the derivations $A' : u_1 \to^* u_n[v_1]_p$ and $B' : v_1 \to^* v_{k+1}$. Both derivations do not contain rewrite steps in non-left-linear variable arguments. Therefore, the rewrite steps

$$u_n[v_1]_p \to_{p.q_1} u_n[v_2]_p \to_{p.q_2} \cdots \to_{p.q_k} u_n[v_{k+1}]_p$$

are nested in at most $d-1$ non-left-linear variable arguments so that for the following derivation the inductive hypothesis for d holds:

$$u_1 \to^* u_n[v_1]_p \to^* u_n[v_{k+1}]_p = u_{n+1}$$

□

Hence, we can eliminate all rewrite steps in non-left-linear variable arguments. We finally eliminate all U-terms with multiple one-step descendants by repeatedly applying the shift-behind.

Lemma 34. *Let $A : u \to^* v, u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly-left-linear DCTRS. Then, there is a derivation $A' : u \to^* v$ such that all U-terms have at most one one-step descendant.*

Proof. We assume that the derivation $A : u_1 \to_{p_1} u_2 \to_{p_2} \cdots u_{n+1}$ does not contain rewrite steps in non-left-linear variable arguments because of Lemma 33.

We prove by induction on the number of U-terms with more than one one-step descendant that there is an equivalent derivation such that all U-terms have at most one one-step descendant.

If there are no such U-terms in A we are finished. Otherwise, let $u_m|_p$ be the first in A and in case of nested U-terms innermost U-term such that $|P| > 1$ where P is the set of one-step descendants of $u_m|_p$.

By Lemma 32 there are derivations $l\sigma \to^* u_m|_p$ and $u_1 \to^* u_{m+1}[l\sigma]_P$ where l is the left-hand side of the corresponding conditional rule.

The derivation $l\sigma \to^* u_m|_p$ only contains U-terms with at most one one-step descendants because it only consists of rewrite steps in $u_1 \to^* u_m$. Since the root symbol of $l\sigma$ is not a U-symbol the following derivation contains less U-terms with multiple one-step descendants than A:

$$u_1 \to^* u_{m+1}[l\sigma]_P \Vdash^* u_{m+1}[u_m|_p]_P \to^* u_{n+1}$$

We therefore can apply the inductive hypothesis to this derivation. □

After eliminating all rewrite steps that duplicate U-terms we can use our previous results to prove soundness:

Lemma 35 (soundness of weakly left-linear DCTRSs)**.** *Let $u \to^* v$, $u \in \mathcal{T}$ be a derivation in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ where $\mathcal{R}$ is a weakly left-linear DCTRS. Then, $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$.*

Proof. By Lemma 34 we obtain a derivation $A' : u \to^* v$ in $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ such that all U-terms have at most one one-step descendant and therefore also at most one non-junk one-step descendant. This implies that all U-terms are either fully eliminated or erased w.r.t. junk terms.

By Lemma 26 and Lemma 30 there is a derivation $u \to^*_{\mathcal{R}} \mathsf{tb}(v)$. □

Using this new proof we can repeat the result of [8, Theorem 31]

Theorem 8 (soundness of weak left-linear DCTRSs for $\mathbb{U}_{\mathsf{seq}}$)**.** *$\mathbb{U}_{\mathsf{seq}}$ is sound for weakly left-linear DCTRSs.*

Proof. Straightforward from Lemma 35. □

Since we obtain soundness w.r.t. the backtranslation tb we also obtain soundness w.r.t. joinability:

Theorem 9 (soundness of weak left-linear DCRTSs for joinability)**.** *$\mathbb{U}_{\mathsf{seq}}$ is sound w.r.t. joinability for weakly left-linear DCTRSs.*

Proof. Straightforward from Lemma 35 and Lemma 1. □

The proof of Theorem 8 differs significantly from the proof that we gave in [8]. In fact this proof resembles the approach that we used to prove soundness of derivations that satisfy the membership condition.

In the proof of Theorem 8 we shift rewrite steps in order to obtain a U-right-linear derivation. In [8] we also need to shift rewrite steps and in fact we use the same shift steps. Yet, we only eliminate rewrite steps in non-left-linear variable arguments and then use the same proof structure that is already used in [29] to proof soundness of left-linear normal 1-CTRSs.

4.8 Confluence of CTRSs

One main motivation to develop transformations is to prove properties of conditional rewrite systems using the well-investigated framework of unconditional rewriting.

Proving or disproving confluence of CTRSs using transformations was already a motivation for the transformation of [4]. This paper presents counterexamples for confluence properties using a simple transformation.

In [9] we presented first results for confluence of CTRSs using unravelings. In order to prove confluence of a conditional rewrite system via the transformed system we need to prove that if two terms are joinable in the transformed system then they are also joinable in the original system.

In general, soundness w.r.t. joinability is essential to prove confluence:

Lemma 36 (confluence and soundness)**.** *$\mathcal{R}$ is confluent if $\mathbb{U}(\mathcal{R})$ is confluent and sound w.r.t. joinability.*

Proof. Let $t_1, t_2 \in \mathcal{T}$ be two terms in the original CTRS $\mathcal{R}$ such that $t_1 \leftrightarrow^*_{\mathcal{R}} t_2$. By completeness of $\mathbb{U}$ we obtain $t_1 \leftrightarrow^*_{\mathbb{U}(\mathcal{R})} t_2$. Since $\mathbb{U}(\mathcal{R})$ is confluent this implies $t_1 \downarrow_{\mathbb{U}(\mathcal{R})} t_2$. By soundness w.r.t. joinability then also $t_1 \downarrow_{\mathcal{R}} t_2$. □

Using this previous lemma we now can prove confluence for many DCTRSs provided that the unraveled DCTRS is confluent:[6]

Lemma 37 (confluence for left-separated DCTRSs)**.** *Let $\mathcal{R}$ be a left-separated DCTRS such that $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is confluent, then $\mathcal{R}$ is confluent.*

Proof. Straightforward from Theorem 3 and Lemma 36. □

Lemma 38 (confluence for weakly left-linear DCTRSs)**.** *Let $\mathcal{R}$ be a weakly left-linear DCTRS such that $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is confluent, then $\mathcal{R}$ is confluent.*

Proof. Straightforward from Theorem 9 and Lemma 36. □

Lemma 39 (confluence for weakly right-linear and sorted DCTRSs)**.** *Let $\mathcal{R}$ be a weakly right-linear and sorted DCTRS such that $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ is confluent, then $\mathcal{R}$ is confluent.*

Proof. Straightforward from Theorem 7 and Lemma 36. □

Since there are automated tests to prove confluence of unconditional term rewrite systems we can use them directly to prove confluence of CTRSs. Since Lemma 36 does not depend on a specific transformation we can also prove soundness w.r.t. joinability for other transformations that have better properties w.r.t. confluence, like e.g. the unraveling of [9].

[6] Erratum 2014-02-10: The properties of the DCTRSs in the following lemmata has been updated compared to the printed version.

CHAPTER 5

Conclusion

5.1 Related Work

5.1.1 Comparison to Other Transformations

Many different transformations have been defined in the past. The first work that specifically addresses properties of transformations known to us is [14]. In this paper the class of unravelings is introduced. Transformations of this class keep the original signature but add new function symbols to encode conditional arguments in which the condition is evaluated.

This first unraveling in [14] has been extended in [15] to the case of deterministic CTRSs in which certain extra variables are allowed. In [28] and [29] this unraveling is slightly refined. These unravelings encode the conditions sequentially, hence we refer to them as *sequential unraveling* in this thesis ($\mathbb{U}_{\mathsf{seq}}$).

Based on the unraveling of [29], another unraveling is introduced in [24]. This optimized sequential unraveling $\mathbb{U}_{\mathsf{opt}}$ only encodes variable bindings if they are used in conditions that are not yet evaluated or the right-hand side of the conditional rule.

Another class of transformations stem from [35]. Based on this approach transformations have been presented in [1], [32] and [6]. In these transformations the original signature is modified: Some function symbols are replaced by function symbols with higher arity. In these additional arguments the condition is encoded and evaluated. Since in these transformations the original symbols are kept, rules that are overlapping in the original CTRS are transformed into overlapping rules. Therefore, these transformations show a better behaviour towards properties like confluence or computational equivalence ([32]).

Although all these transformations split conditional rules into multiple unconditional ones in a similar way we obtain different soundness properties. In the following we want to present counterexamples to show that soundness of $\mathbb{U}_{\mathsf{seq}}$ does not imply soundness for other transformations in general.

In [25] it is shown that soundness of $\mathbb{U}_{\mathsf{opt}}$ is sufficient for soundness of $\mathbb{U}_{\mathsf{seq}}$. The reverse of this result, soundness of $\mathbb{U}_{\mathsf{seq}}$ implies soundness of $\mathbb{U}_{\mathsf{opt}}$, does not hold: We have pointed out in

[7] that $\mathbb{U}_{\mathsf{opt}}$ gives rise to unsoundness for rewrite sequences in which U-terms are erased (see also Example 5). In the optimized unraveling terms terms in which conditions are evaluated encode less information than the term that initiated the conditional evaluation. We therefore may obtain joinability for terms in the transformed system that are not joinable in the original system, in particular if a condition is not satisfied. Combined with non-left-linear rules this leads to unsoundness.

Yet in these unsound derivations the terms that encode the conditions must be erased. Otherwise they remain as witnesses for an unsuccessful conditional evaluation and cannot be removed. Therefore, in non-erasing systems these garbage terms are kept, hence although we still obtain unsoundness for joinability such terms are preserved in derivations so that we do not obtain a term of the original conditional term rewrite system.

Yet, also for $\mathbb{U}_{\mathsf{opt}}$-non-erasing systems we obtain unsoundness even if $\mathbb{U}_{\mathsf{seq}}$ is sound:

Example 41. *Consider the following DCTRS and its unraveling using $\mathbb{U}_{\mathsf{opt}}$:*

$$\mathcal{R} = \left\{ \begin{array}{rl} s(a) & \searrow \\ & c \nearrow t(k) \\ s(b) & \nearrow \\ & \searrow t(l) \\ f(x) \to z & \Leftarrow s(x) \to^* t(z) \\ g(x,x) \to h(x,x) & \end{array} \right\} \quad \mathbb{U}_{\mathsf{opt}}(\mathcal{R}) = \left\{ \begin{array}{rl} s(a) & \searrow \\ & c \nearrow t(k) \\ s(b) & \nearrow \\ & \searrow t(l) \\ f(x) & \to U_1^\alpha(s(x)) \\ U_1^\alpha(t(z)) & \to z \\ g(x,x) & \to h(x,x) \end{array} \right\}$$

Consider the following derivation in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$

$$\begin{aligned} g(f(a),f(b)) &\to g(U_1^\alpha(s(a)),U_1^\alpha(s(b))) \Vdash g(U_1^\alpha(c),U_1^\alpha(c)) \\ &\to h(U_1^\alpha(c),U_1^\alpha(c)) \Vdash h(U_1^\alpha(t(k)),U_1^\alpha(t(l))) \Vdash h(k,l) \end{aligned}$$

This derivation is unsound because $f(a)$ and $f(b)$ do not have a common reduct that rewrites to k and l. The derivation is therefore unsound although the CTRS is $\mathbb{U}_{\mathsf{opt}}$-non-erasing.

Consider the unraveled TRS using $\mathbb{U}_{\mathsf{seq}}$:

$$\mathbb{U}_{\mathsf{seq}}(\mathcal{R}) = \left\{ \begin{array}{rl} s(a) & \searrow \\ & c \nearrow t(k) \\ s(b) & \nearrow \\ & \searrow t(l) \\ f(x) & \to U_1^\alpha(s(x),x) \\ U_1^\alpha(t(z),x) & \to z \\ g(x,x) & \to h(x,x) \end{array} \right\}$$

Now, we cannot apply the non-linear rule because there is no common reduct of a and b:

$$g(f(a),f(b)) \to g(U_1^\alpha(s(a),a),U_1^\alpha(s(b),b)) \Vdash g(U_1^\alpha(c,a),U_1^\alpha(c,b))$$

In fact, a and b are irreducible. Every rewrite sequence in $\mathbb{U}_{\mathsf{seq}}$ starting from $g(f(a),f(b))$ therefore satisfies the soundness condition for context-sensitive rewriting.

Other transformations are usually only defined for normal 1-CTRSs ([1], [32] and [6]). These transformations replace existing function symbols by symbols with a higher arity. Therefore, terms encoding conditional arguments still resemble original terms, yet since the transformed system changed some function symbols they are not covered by the original definition of unravelings of [14]. Still, these transformations resemble the unraveling $\mathbb{U}_{\mathsf{seq}}$ because they keep all variable bindings of the term that starts the conditional evaluation (in fact the whole term is kept). We introduced a framework to describe properties of transformations of CTRSs into TRSs in [6] that covers these transformations and also unravelings.

It is surprising that soundness properties of $\mathbb{U}_{\mathsf{seq}}$ do not carry over to these other transformations, even if we do not consider CTRSs as in Example 10. Even for normal 1-CTRSs, we obtain different soundness results for the transformation of [1] and therefore also for the transformations of [32] or [6].

Example 42. *Consider the following normal 1-CTRS and its transformed TRS using the transformation of [1]:*

$$\mathcal{R} = \left\{ \begin{array}{r} a \to c \\ \diagup\!\!\diagdown \\ b \to d \\ f(x) \to x \Leftarrow x \to^* c \\ g(x,x) \to h(x,x) \\ h(x, f(x)) \to x \end{array} \right\} \qquad \mathbb{T}_{[1]}(\mathcal{R}) = \left\{ \begin{array}{r} a \to c \\ \diagup\!\!\diagdown \\ b \to d \\ f'(x,\bot) \to f'(x,\langle x\rangle) \\ f'(x,\langle c\rangle) \to x \\ g(x,x) \to h(x,x) \\ h(x, f'(x,z)) \to x \end{array} \right\}$$

The root symbol of the left-hand side of the conditional rule f is replaced by a new function symbol f' with an additional argument to encode the conditional argument. If the conditional argument is not initialized it contains $\bot$. The term $g(f(a), f(b))$ therefore corresponds to the term $g(f'(a,\bot), f'(b,\bot))$ in the transformed TRS.

In all rules we need to replace f-terms by f'-terms. In order to avoid that partial conditional evaluations block derivations the conditional argument is mapped to a new variable and therefore ignored.

Now consider the following derivation:

$$\begin{aligned} g(f'(a,\bot), f'(b,\bot)) &\to^* g(f'(a,\langle a\rangle), f'(b,\langle b\rangle)) \to^* g(f'(d,\langle c\rangle), f'(d,\langle c\rangle)) \\ &\to h(f'(d,\langle c\rangle), f'(d,\langle c\rangle)) \to h(d, f'(d,\langle c\rangle)) \to d \end{aligned}$$

We already explained in Example 21 that the derivation $g(f(a), f(b)) \to^* h(d, f(d))$ is not possible in $\mathcal{R}$. Therefore, also the above derivation is unsound. Therefore, $\mathbb{T}_{[1]}$ is unsound in this case although $\mathbb{U}_{\mathsf{opt}}$ is sound by Theorem 1.

Since the transformation of [1] is not sound for non-overlay systems (see Example 10) and the refinements in [32] and [6] are syntactically very complex we introduced an unraveling in [9] that combines some properties of both classes of transformations.

This unraveling $\mathbb{U}_{\mathsf{conf}}$ shows better properties w.r.t. confluence for overlay CTRSs in which conditional rules depend on similar rules, yet it still satisfies the original definition of unravelings. Such overlaying rules arise very often when modeling pattern matching of functional

programming languages similar to in Example 3. This new unraveling is already hinted in [29, Example 7.2.49].

In this unraveling we do not use the rule as a label for U-symbols but only the left-hand side of the condition and the rule (independent of a renaming of the variable). We also obtain different properties w.r.t. soundness for this transformation:

Example 43. *Consider the following $\mathbb{U}_{\mathsf{opt}}$-non-erasing 1-CTRS and its transformation using the unraveling of [9]:*

$$\mathcal{R} = \left\{ \begin{array}{l} f(x) \to A \Leftarrow s(x) \to^* t \\ f(x) \to B \Leftarrow s(x) \to^* t \\ \begin{array}{r} s(a) \\ s(b) \end{array} \searrow\nearrow t \\ \begin{array}{r} a \\ b \end{array} \searrow\nearrow c \\ g(x,x) \to h(x,x) \end{array} \right\} \quad \mathbb{U}_{\mathit{new}}(\mathcal{R}) = \left\{ \begin{array}{r} f(x) \to U_{f(x),s(x)}(s(x),x) \\ U_{f(x),s(x)}(t,x) \to A \\ U_{f(x),s(x)}(t,x) \to B \\ \begin{array}{r} s(a) \\ s(b) \end{array} \searrow\nearrow t \\ \begin{array}{r} a \\ b \end{array} \searrow\nearrow c \\ g(x,x) \to h(x,x) \end{array} \right\}$$

Observe that we obtain one rule less in $\mathbb{U}_{\mathit{new}}(\mathcal{R})$ than in $\mathbb{U}_{\mathsf{opt}}(\mathcal{R})$ or $\mathbb{U}_{\mathsf{seq}}(\mathcal{R})$ because the introduction rules of both conditional rules are identical.

In the unraveled CTRS we obtain the following derivation:

$$\begin{aligned} g(f(a),f(b)) &\to^* g(U_{f(x),s(x)}(s(a),a), U_{f(x),s(x)}(s(b),b)) \\ &\to^* g(U_{f(x),s(x)}(t,c), U_{f(x),s(x)}(t,c)) \\ &\to^* h(U_{f(x),s(x)}(t,c), U_{f(x),s(x)}(t,c)) \to^* h(A,B) \end{aligned}$$

Yet, this derivation is unsound because there is no common descendant of $f(a)$ and $f(b)$ that rewrites to A and B in $\mathcal{R}$.

It will be part of our future work to provide soundness results also for these transformations.

5.1.2 Applications

Since conditional term rewrite systems are more complex than unconditional ones there has been many efforts to adapt properties and results of unconditional term rewrite systems using transformations. For this purpose, soundness properties are essential. We already presented some confluence results for CTRSs that may lead to automated confluence proofs for conditional term rewrite systems. In the recent literature there are further applications of soundness results.

Inversion of Functions

In [17] a method is shown to invert functions by inverting the term rewrite system that represents the function. The method returns a conditional rewrite system. In order to eliminate the conditions of these CTRSs unravelings are used.

In [24] this method is investigated in more detail, along with some soundness results. Unravelings are used in [24] to (partially) invert functions. In [21] this method is also used to prove injectivity of functions, and in [26] inversion is presented for tail-recursive functions.

In this inversion approach unravelings are used. The analysis of soundness is therefore of major interest and addressed to in [25]. Although the authors focus on the optimized unraveling their results immediately carry over to the sequential unraveling as it is shown in [25].

Operational Termination of CTRSs

In [31] it is shown that we obtain soundness if we use context-sensitive rewriting in the transformed system. Based on this result new results on operational termination of CTRSs are proven, for instance it is shown that non-termination of the transformed system implies non-operational termination. Using unravelings, provers for termination for unconditional term rewrite systems can be used to prove (operational) termination of conditional term rewrite systems.

Computational Equivalence

In [1] a transformation is introduced that is based on the transformation of [35]. The main purpose of this transformation is to simplify conditional narrowing and to allow parallel evaluation of conditions. The transformation itself has better properties for preserving confluence than unravelings, yet it also is unsound for many non-overlay systems.

In [32] a new property for transformations is introduced, computational equivalence. This property implies that the transformed TRS of a confluent CTRS is confluent and sound. We called this property *soundness for preserving normalforms* in [6]. The benefit of this property is that we can use the transformed TRS to simulate all normalizing conditional rewrite sequences and thereby use unconditional rewrite engines for the conditional case. The usefulness of this approach is shown by comparing various experimental results of transformed CTRSs. Nonetheless the transformed TRS are syntactically complex (see Example 42). For instance, they do not preserve important properties like being a constructor system or being an overlay system.

In [6] we therefore investigated in detail why the transformation of [1] is unsound for certain CTRSs. This lead to the definition of a transformation that also is computationally equivalent for many CTRSs (yet not for all cases in which [32] is computationally equivalent) and has better syntactical properties.

5.2 Summary

Conditional term rewrite systems arise naturally in many applications, yet many notions of unconditional rewriting change their intuitive meaning and many criteria for proving properties like confluence or termination are not valid after adding conditions. For this reason, transforming conditional systems into unconditional ones is an intuitive approach to analyze conditional rewrite systems.

In order to prove properties of conditional term rewrite systems using transformations into unconditional ones we need to know whether rewrite sequences in the original system correspond to rewrite sequences in the transformed system. This property is called completeness

(for reducibility). It is usually satisfied because transformations add some additional symbols to encode the conditions and are therefore more powerful than conditional rewrite systems.

It is much harder to prove whether rewrite sequences in the transformed system correspond to rewrite sequences in the original system. This property is called soundness (for reducibility). Transformations are not sound in general because the increased power may give rise to new derivations in the transformed system. Yet, in order to benefit from the well-understood framework of unconditional rewriting for conditional rewrite systems soundness (or a similar property) is essential.

There are many practical and theoretical applications for soundness properties, some of which are the following:

- In order to prove that a conditional term rewrite system is not operationally terminating we need to prove that an infinite rewrite sequence in the transformed system corresponds to a computable infinite rewrite sequence in the original system. This is investigated in detail in [31].

- In order to prove confluence of conditional term rewrite systems using transformations we need to prove joinability of terms. For this purpose we need to translate rewrite sequences in the transformed system into conditional rewrite sequences (see Lemma 36 and [9]).

- From a practical point of view, implementing conditional term rewriting is far from trivial. Transformations can be used to benefit from features of unconditional rewriting. In [32] the notion of computational equivalence is introduced for this purpose. We investigated this approach in [6] along with other transformations.

- Another practical application is given by the fact that (deterministic) conditional term rewrite systems are an intuitive way to encode functional programs. Therefore, any kind of syntactic transformation of functional programs can be implemented in conditional rewriting. [24] provides such a transformation in order to invert functions. In the last step of the inversion the conditional rewrite system is unraveled. For this purpose, unravelings are used.

In [14], a class of transformations, unravelings, is introduced. It is shown that a certain unraveling is not sound (see Example 20). Nonetheless it is shown that the unraveling is sound for left-linear normal 1-CTRSs, a simple class of conditional term rewrite systems. In [15], [27] and [29] a refinement of this unraveling is presented that is also applicable to conditional rewrite systems with deterministic extra variables. Based on this transformation, [24] presents an unraveling that optimizes the use of variables. [25] proves soundness for left-linear deterministic conditional rewrite systems, and soundness for non-erasing and right-linear deterministic conditional rewrite systems for this unraveling. Furthermore it is shown that soundness of this optimized unraveling implies soundness for the unraveling of [29].

We investigated soundness properties in [7] for normal 1-CTRSs, based on the unraveling of [14] and proved and disproved many new results. We proved soundness also for a class of non-left-linear systems, so-called weakly left-linear systems. In weakly left-linear conditional term

rewrite systems we allow non-left-linear rules like $eq(x, x) \to true$ in which non-left-linear variables are erased.

In [8] we extended our analysis and results to deterministic conditional term rewrite systems and obtained many new results, yet we also could prove that many results of [7] do not hold in presence of extra variables. Parts of this thesis are based on our results in [8].

In this thesis we first motivate the use of unravelings and sketch reasons for unsoundness in Chapter 1. In conditional rewriting the evaluation of conditions is separated from other rewrite steps and only if the conditions are satisfied we continue rewriting of the term that initiated the conditional evaluation. Therefore, the term that initiated the conditional evaluation and the conditional evaluation itself are strictly separated and synchronized.

In unconditional rewriting, conditional evaluations are encoded along with the variable bindings of the term that initiated the conditional evaluation. Therefore, we can continue rewriting inside the variables of this term and therefore obtain rewrite sequences in which the conditional argument is outdated because of further rewrite steps in the variable bindings.

One way to avoid such inconsistencies is to use a strategy that avoids that the variable bindings can be rewritten. This is sufficient for soundness as is shown in [31]. Yet, in general it is not possible to avoid such inconsistencies. We therefore focus in this thesis which properties imply that there either are no such inconsistencies or under which circumstances they do not cause unsoundness.

In Chapter 3 we repeat the definition of the unravelings of [14] (Definition 1), [29] (Definition 2) and [24] (Definition 3). In all unravelings conditions are encoded in terms that are rooted by a new function symbol, along with some variable bindings. We then define translations of such U-terms into terms in the original system. Such term translations are used to map terms that represent partly evaluated conditions in the transformed system into terms in the original system. We discuss two possibilities:

- The backtranslation `tb` maps a partly evaluated condition to the corresponding left-hand side of the conditional rule. This corresponds to the assumption that the condition might not be satisfied.
- The translation forward `tf` maps a U-term to the corresponding right-hand side. We use this translation when we know that the condition is satisfied.

In Chapter 4, our main chapter, we first define soundness properties. Since the literature is not consistent in notions of soundness we introduced a unified framework for transformations of CTRSs into TRSs in [6]. Our soundness notions are based on this paper.

One reason for unsoundness is that rewrite steps in the variable bindings in a U-term might flip the status of satisfiability of the conditions. Yet, by imposing some restriction on how variables are distributed in the conditions we can avoid such cases of unsoundness (see Example 18).

We called the corresponding property sortedness (Definition 11). Sortedness is important to avoid unsoundness, yet it is not sufficient to obtain soundness. This is shown in some counterexamples.

In order to prove soundness of sorted CTRSs we then presented two different approaches. In the first approach we translate derivations in the transformed system into derivations in the original system using term translations. We then prove that right-separateness and non-erasingness

are sufficient for soundness of 2-DCTRSs (Theorem 1). This class of CTRSs does not allow extra variables on the right-hand side of the conditional rules but they are allowed inside the conditions. Nonetheless, we do not obtain other soundness properties like soundness w.r.t. joinability.

The symmetric counterpart to this result is soundness for left-separate DCTRSs (Theorem 2). Yet since this property is syntactically very strict this property is of limited practical use. Similar to the last property we prove soundness w.r.t. joinability for confluent CTRSs. This case is interesting because we do not obtain soundness.

For other cases we cannot directly translate derivations of the transformed system into conditional rewrite sequences. Instead, we introduce a new class of derivations, so-called U-eager derivations, and prove that they are sound..

Using this approach we prove that membership rewriting is sufficient for soundness of the optimized unraveling, provided that the CTRS is sorted (Theorem 5). This result is remarkable because in [20] it is shown that context-sensitivity and membership rewriting are sufficient for soundness. We even can weaken the membership condition and restrict it to non-right-linear variables, i.e., variables that have more than one one-step descendant. This last result is then the source for soundness proofs for further soundness results in the sequential unraveling.

For this purpose we present a mapping from derivations in the sequential unraveling into the optimized sequential unraveling, $\mathsf{t}_{\mathsf{opt}}$ (Definition 18). Using this mapping we show that weak right-linearity (Definition 19) is sufficient for soundness. This is an important improvement of other soundness results: In [7] we have shown that we obtain soundness if the transformed system is right-linear (using the simultaneous unraveling). This result actually resembles our result on left-separate CTRSs. In [25] it was shown that we obtain soundness for right-linear and non-erasing systems.

We then repeat a result of [8] and prove that we also obtain soundness for weakly left-linear DCTRSs (see Definition 20). The proof of this result (Theorem 8) is simpler than the one in [8].

Finally we show that we also obtain soundness w.r.t. joinability in both cases and show that we can prove confluence of CTRSs by proving soundness w.r.t. joinability and confluence of the transformed system. We published a similar result in [9].

Summarizing, we have shown several new results for soundness and disproven many other. First we provided many counterexamples for soundness for the unravelings of $\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$.

- Example 5: $\mathbb{U}_{\mathsf{opt}}$ is unsound for erased CTRSs. This example also disproves soundness w.r.t. joinability of $\mathbb{U}_{\mathsf{opt}}$ for most CTRSs.
- Example 18: $\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$ are unsound for non-sorted CTRSs.
- Example 20: Counterexample for soundness of [14].
- Example 21: In-depth analysis of unsoundness example.
- Example 22: Example for unsoundness of sorted non-erasing DCTRSs from [8]. Using a similar CTRS in Example 23 we even disprove soundness for confluent sorted non-erasing DCTRSs.

In [7] soundness is shown by translating mixed terms into original terms. Using this approach we proved several results for normal 1-CTRSs already in [7] and extended them to DCTRSs [8].

- Theorem 1: Soundness of $\mathbb{U}_{\mathsf{opt}}$ for right-separate, $\mathbb{U}_{\mathsf{opt}}$-non-erasing 2-DCTRSs. Example 21 disproves soundness w.r.t. joinability in this case. This result is not valid for many other transformations including the unraveling of [9] (Example 41).

- Theorem 2: Soundness for left-separated DCTRSs. We also obtain soundness w.r.t. joinability (Theorem 3). This case is an extension of our result for ultra-right-linear CTRSs in [7] and [8].

- Theorem 4: $\mathbb{U}_{\mathsf{seq}}$ is sound w.r.t. joinability for confluent, right-stable CTRSs. We also have shown this result already in [8]. In [7] we even obtain soundness for normal 1-CTRSs. For DCTRSs, Example 23 is a counterexample.

Translating derivations showed its limitations for more complex syntactic properties. We therefore use a proof approach that is already hinted in [8] to proof further soundness properties: We define a class of derivations of which we know that they are sound and show that we can convert derivations of certain CTRSs into such derivations, thus proving soundness.

In [29] and [14] so-called balanced reductions are used for this case. We use a different class of derivations, so-called U-eager derivations. This strategy can be described as "U-terms first", meaning that if we introduce a term that encodes a condition the condition must be satisfied and eliminated before we may apply other rewrite steps. We prove soundness for sorted CTRSs of such U-eager derivations in Lemma 14 (for $\mathbb{U}_{\mathsf{opt}}$). Based on this result we obtain further soundness results:

- Theorem 5: Soundness of $\mathbb{U}_{\mathsf{opt}}$ for derivations that satisfy the membership condition $\in \mathcal{T}$ for sorted CTRS. This result is interesting because in [18] it is shown that $\mathbb{U}_{\mathsf{opt}}$ is sound for context-sensitive derivations that satisfy the membership condition. In [31] it is shown that $\mathbb{U}_{\mathsf{seq}}$ is sound for context-sensitive derivations, yet Example 29 shows that $\mathbb{U}_{\mathsf{opt}}$ is not sound for such derivations, even for sorted CTRSs.

- Lemma 19: Soundness of $\mathbb{U}_{\mathsf{opt}}$ for derivations in which all U-terms have at most one one-step descendant and no U-term is erased. One implication of this result has been shown already in [23], that is soundness of $\mathbb{U}_{\mathsf{opt}}$ for non-erasing, $\mathbb{U}_{\mathsf{opt}}$-right-linear CTRSs.

- Lemma 27: Soundness of $\mathbb{U}_{\mathsf{seq}}$ for derivations in which all U-terms have at most one (non-junk) one-step descendant. This result is an extension of Lemma 19 to $\mathbb{U}_{\mathsf{seq}}$. Since $\mathbb{U}_{\mathsf{seq}}$ encodes more information than $\mathbb{U}_{\mathsf{opt}}$ we can drop the requirement for non-erasingness of U-terms. Furthermore, the overhead of the encoding (that we called *junk terms*) can be ignored.

- Theorem 6: Soundness of $\mathbb{U}_{\mathsf{seq}}$ for weakly right-linear, sorted DCTRSs. For this class of CTRSs we also obtain soundness w.r.t. joinability (Theorem 7).

- Theorem 8: Soundness of $\mathbb{U}_{\mathsf{seq}}$ for weakly left-linear DCTRSs. Here we used a different proof approach than in [8]. Weak left-linearity also implies soundness w.r.t. joinability (Theorem 9).

In this thesis we mostly discuss soundness the sequential unravelings $\mathbb{U}_{\mathsf{opt}}$ and $\mathbb{U}_{\mathsf{seq}}$. There are some examples that show that we cannot simply adapt all soundness results to other transformations, most notably transformations stemming from [1].

These transformations have some advantages concerning confluence properties because they do not block derivations when a condition is evaluated. It therefore is of some interest to investigate in which cases our results of this thesis carry over to other transformations.

A related topic to proving properties of conditional rewrite systems using transformations is reachability analysis: Not all terms in the transformed system represent a valid state of conditions because they are not reachable from original terms. Nonetheless they lead to non-confluence or non-operational termination ([29, Example 7.2.51]). We therefore should not consider such unreachable terms ([6]) in the analysis of properties.

Of major interest is also the possibility to implement conditional rewriting using transformations. In [32] some experimental results are provided, and in [6] we compare two transformations.

From a practical point of view, our soundness results may also lead to new applications of syntactic transformations of CTRSs. Inversion of functions has been a subject of interest for some time (e.g. [24], [26]) and we hope that our work contributes to ongoing efforts in this area of research.

Another practical application is the implementation of conditional term rewriting. Using transformations we can benefit from current efficient implementations. Related to this we can use transformations to prove confluence of conditional rewrite systems automatically. We hope that this thesis will be a useful contribution for this purpose.

Bibliography

[1] Sergio Antoy, Bernd Brassel, and Michael Hanus. Conditional narrowing without conditions. In *Proc. 5th International ACM SIGPLAN Conference on Principles and Practice of Declarative Programming, 27-29 August 2003, Uppsala, Sweden*, pages 20–31. ACM Press, 2003.

[2] Jürgen Avenhaus and Carlos Loría-Sáenz. On conditional rewrite systems with extra variables and deterministic logic programs. In Frank Pfenning, editor, *Proc. 5th Int. Conf. on Logic Programming and Automated Reasoning (LPAR'94), Kiev, Ukraine, July 16-22, 1994*, pages 215–229, 1994.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and All That*. Cambridge University Press, 1998.

[4] Jan A. Bergstra and Jan Willem Klop. Conditional rewrite rules: Confluence and termination. *Journal of Computer and System Sciences*, 32(3):323–362, 1986.

[5] Elio Giovanetti and Corrado Moiso. Notes on the elimination of conditions. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Proc. 1st Int. Workshop on Conditional Rewriting Systems (CTRS'87), Orsay, France, 1987*, volume 308 of *Lecture Notes in Computer Science*, pages 91–97, Orsay, France, 1988. Springer. ISBN 3-540-19242-5.

[6] Karl Gmeiner and Bernhard Gramlich. Transformations of conditional rewrite systems revisited. In Andrea Corradini and Ugo Montanari, editors, *Recent Trends in Algebraic Development Techniques (WADT 2008) – Selected Papers*, volume 5486 of *Lecture Notes in Computer Science*, pages 166–186. Springer, 2009.

[7] Karl Gmeiner, Bernhard Gramlich, and Felix Schernhammer. On (un)soundness of unravelings. In Christopher Lynch, editor, *Proc. 21st International Conference on Rewriting Techniques and Applications (RTA 2010), July 11-13, 2010, Edinburgh, Scotland, UK*, LIPIcs (Leibniz International Proceedings in Informatics), July 2010.

[8] Karl Gmeiner, Bernhard Gramlich, and Felix Schernhammer. On soundness conditions for unraveling deterministic conditional rewrite systems. In Ashish Tiwari, editor, *Proc. 23rd International Conference on Rewriting Techniques and Applications (RTA 2012), May 30 – June 2, 2012, Nagoya, Japan*, LIPIcs (Leibniz International Proceedings in Informatics), May/June 2012.

[9] Karl Gmeiner, Naoki Nishida, and Bernhard Gramlich. Proving confluence of conditional term rewriting systems via unravelings. In Nao Hirokawa and Vincent van Oostrom, editors, *Proceedings of the 2nd International Workshop on Confluence*, pages 35–39, 2013.

[10] Claus Hintermeier. How to transform canonical decreasing ctrss into equivalent canonical trss. In *Conditional and Typed Rewriting Systems, 4th International Workshop, CTRS-94, Jerusalem, Israel, July 13-15, 1994, Proceedings*, volume 968 of *Lecture Notes in Computer Science*, pages 186–205, 1995.

[11] S. Kaplan. Conditional rewrite rules. *Theor. Comput. Sci.*, 33(2-3):175–193, 1984.

[12] Salvador Lucas, Y. Guo, and M. Hagiya. Context-sensitive computations in functional and functional logic programming, 1998.

[13] Salvador Lucas, Claude Marché, and José Meseguer. Operational termination of conditional term rewriting systems. *Inf. Process. Lett.*, 95(4):446–453, 2005.

[14] Massimo Marchiori. Unravelings and ultra-properties. In Michael Hanus and Mario Mario Rodríguez-Artalejo, editors, *Proc. 5th Int. Conf. on Algebraic and Logic Programming, Aachen*, volume 1139 of *Lecture Notes in Computer Science*, pages 107–121. Springer, September 1996.

[15] Massimo Marchiori. On deterministic conditional rewriting. Technical Report MIT LCS CSG Memo n.405, MIT, Cambridge, MA, USA, October 1997.

[16] Aart Middeldorp and Eric Hamoen. Completeness results for basic narrowing. *Applicable Algebra in Engineering, Communication and Computing*, 5(3–4):213–253, 1994.

[17] Naoki Nishida. *Transformational Approach to Inverse Computation in Term Rewriting*. PhD thesis, Nagoya University, Nagoya, Japan, 2004.

[18] Naoki Nishida, Tomohiro Mizutani, and Masahiko Sakai. Transformation for refining unraveled conditional term rewriting systems. *Electr. Notes Theor. Comput. Sci. (ENTCS)*, 174(10), 2007. Sergio Antoy, ed., Final Proc. 6th International Workshop on Reduction Strategies in Rewriting and Programming.

[19] Naoki Nishida, Masahiko Sakai, , and Toshiki Sakabe. On simulation-completeness of unraveling for conditional term rewriting systems. *IEICE Technical Report SS2004-18*, 104(243):25–30, 2004. Revised version from December 27, 2005, 15 pages.

[20] Naoki Nishida, Masahiko Sakai, , and Toshiki Sakabe. On simulation-completeness of unraveling for conditional term rewriting systems. In *LA Symposium 2004-7 Summer*, pages 71–76, 2004.

[21] Naoki Nishida and Masahiko Sakai. Completion as post-process in program inversion of injective functions. In Aart Middeldorp, editor, *Proc. 8th International Workshop on Reduction Strategies in Rewriting and Programming, July 14, 2008, Hagenberg, Austria*, pages 61–75, July 2008.

[22] Naoki Nishida and Masahiko Sakai. Completion after program inversion of injective functions. *Electr. Notes Theor. Comput. Sci.*, 237:39–56, 2009. Proceedings of the 8th International Workshop on Reduction Strategies in Rewriting and Programming (WRS 2008), Castle of Hagenberg, Austria, 14 July 2008, Aart Middeldorp, ed.

[23] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. Narrowing-based simulation of term rewriting systems with extra variables. *Electr. Notes Theor. Comput. Sci.*, 86(3), 2003. Proc. WFLP'03, 12th International Workshop on Functional and Constraint Logic Programming.

[24] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. Partial inversion of constructor term rewriting systems. In Jürgen Giesl, editor, *Proc. 16th International Conference on Rewriting Techniques and Applications (RTA'05), Nara, Japan, April 19-21, 2005*, volume 3467 of *Lecture Notes in Computer Science*, pages 264–278. Springer, April 2005.

[25] Naoki Nishida, Masahiko Sakai, and Toshiki Sakabe. Soundness of unravelings for deterministic conditional term rewriting systems via ultra-properties related to linearity. In Manfred Schmidt-Schauss, editor, *Proc. 22nd International Conference on Rewriting Techniques and Applications (RTA 2011), May 30 – June 1, 2011, Novi Sad, Serbia*, LIPIcs (Leibniz International Proceedings in Informatics), 2011. pages 267–282.

[26] Naoki Nishida and Germán Vidal. Program inversion for tail recursive functions. In Manfred Schmidt-Schauß, editor, *RTA*, volume 10 of *LIPIcs*, pages 283–298. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[27] Enno Ohlebusch. On quasi-reductive and quasi-simplifying deterministic conditional rewrite systems. In Aart Middeldorp and Taisuke Sato, editors, *Proc. 4th Fuji Int. Symp. on Functional and Logic Programming (FLOPS'99)*, volume 1722 of *Lecture Notes in Computer Science*, pages 179–193, Tsukuba, Japan, November 1999. Springer.

[28] Enno Ohlebusch. Transforming conditional rewrite systems with extra variables into unconditional systems. In Harald Ganzinger, David A. McAllester, and Andrei Voronkov, editors, *Proc. 6th Int. Conf. on Logic Programming and Automated Reasoning (LPAR'99)*, volume 1705 of *Lecture Notes in Computer Science*, pages 111–130, Tbilisi, Georgia, September 1999. Springer.

[29] Enno Ohlebusch. *Advanced Topics in Term Rewriting*. Springer, 2002.

[30] Grigore Rosu. From conditional to unconditional rewriting. In José Luiz Fiadeiro, Peter D. Mosses, and Fernando Orejas, editors, *WADT*, volume 3423 of *Lecture Notes in Computer Science*, pages 218–233. Springer, 2004.

[31] Felix Schernhammer and Bernhard Gramlich. Characterizing and proving operational termination of deterministic conditional term rewriting systems. *Journal of Logic and Algebraic Programming*, 79(7):659–688, 2010. Revised selected papers of NWPT 2008, Tarmo Uustalu and Jüri Vain, editors.

[32] Traian-Florin Şerbănuţă and Grigore Roşu. Computationally equivalent elimination of conditions. In Frank Pfenning, editor, *Proc. 17th International Conference on Rewriting Techniques and Applications, Seattle, WA, USA, August 12-14, 2006*, volume 4098 of *Lecture Notes in Computer Science*, pages 19–34. Springer, 2006.

[33] Taro Suzuki, Aart Middeldorp, and Tetsuo Ida. Level-confluence of conditional rewrite systems with extra variables in right-hand sides. In Jieh Hsiang, editor, *Proc. 6th Int. Conf. on Rewriting Techniques and Applications (RTA'95), Kaiserslautern, Germany*, volume 914, pages 179–193, Kaiserslautern, Germany, April 1995. Springer-Verlag.

[34] Yoshihito Toyama. Confluent term rewriting systems with membership conditions. In Stéphane Kaplan and Jean-Pierre Jouannaud, editors, *Proc. 1st Int. Workshop on Conditional Term Rewriting Systems, Orsay, France, July 8-10, 198*, volume 308 of *Lecture Notes in Computer Science*, pages 228–241. Springer, 1988.

[35] Patrick Viry. Elimination of conditions. *J. Symb. Comput.*, 28(3):381–401, 1999.

List of Tables

List of Figures

Index