Equational Reasoning and Completion

by

Dominik Klein

submitted to
Japan Advanced Institute of Science and Technology
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

Supervisor: Professor Mizuhito Ogawa

School of Information Science
Japan Advanced Institute of Science and Technology

June 2012

Abstract

Equations are a versatile and natural tool for various reasoning tasks. This thesis investigates how to fully automate equational reasoning. Central to automation is Knuth and Bendix' ground-breaking completion procedure. It was originally formulated as an algorithm, and requires a user-provided reduction order. We show that completion can be characterized as an optimization problem. Using this formulation, called Maximal Completion, algorithmic aspects can be abstracted away by encoding the optimization problem to constraints. Full automation can be achieved by constructing a suitable order from the solution of the constraint problem.

The success of Knuth and Bendix' procedure has led to several adaptations to related reasoning tasks, such as Inductionless Induction or Rewriting Induction. We propose a new uniform framework based on constrained equalities. It allows to easily formulate several completion procedures by simple parameter changes. But more importantly it also makes it possible to adopt a similar approach as above, namely reformulating these adaptations as optimization problems. A convenient side effect of this framework is that soundness proofs are simplified, since necessary conditions are encoded in the constraint part.

Lastly we investigate confluence of Term Rewriting Systems (TRSs). Testing confluence automatically is a central property not only for completion, but in other applications as well, such as narrowing or type theory. We present a new confluence criterion for non-left-linear and non-terminating TRSs. Since it requires joinability of uncomputable extended critical pairs, we provide a new result on unification, which enables full automation of our criterion.

All results presented in this thesis have been implemented and experimentally evaluated to show their practical effectiveness.

Acknowledgments

I am much indebted to both Associate Professor Nao Hirokawa and Professor Mizuhito Ogawa for supporting me throughout my studies, for their constant encouragement, and for providing valuable advice. Nao Hirokawa introduced me into the world of term rewriting, and supervised large parts of my research. His attitude towards teaching and research has been a great inspiration, and I would like to thank him for his guidance and, an education. I would also like to thank Associate Professor Xavier Défago, who supervised my "minor research project" in the PhD Program, and François Bonnet for his cooperation. I feel very honoured and grateful that all referees, Associate Professor Takahito Aoto, Professor Aart Middeldorp, Associate Professor Kazuhiro Ogata, Professor Michio Oyamaguchi, and Assistant Professor Haruhiko Sato accepted to review this thesis. Professor Aart Middeldorp and Sarah Winkler also provided valuable comments on Maximal Completion, which helped to significantly improve the presentation. Last but not least I would like to thank Hiroko and my family and friends for their moral support.

Contents

Chapter 1

Introduction

This introduction consists of two parts. The first part provides an informal introduction into equational reasoning based on term rewriting, and associated challenges and open problems. The second part gives an overview of the approaches taken to tackle these challenges, and lists the contributions of this thesis.

1.1 Equational Reasoning and its Challenges

To express and reason about various facts, several formalisms exist. Very often, such facts and relationships between them can be very naturally expressed as equations. For example, addition on natural numbers can be axiomatized by the equational system (ES):

$$\begin{aligned} 1\colon &\quad 0 + x \approx x \\ 2\colon &\quad \mathsf{s}(x) + y \approx \mathsf{s}(x + y) \end{aligned}$$

We could now ask whether $1 + 0$ equals $0 + 1$, i.e. whether $\mathsf{s}(0) + 0 \approx 0 + \mathsf{s}(0)$ follows from the given axioms. A proof can be constructed by equational reasoning:

	lhs		**rhs**
	$\mathsf{s}(0)+0$		$0+\mathsf{s}(0)$
$\leftrightarrow_2$	$\mathsf{s}(0+0)$	$\leftrightarrow_1$	$\mathsf{s}(0)$
$\leftrightarrow_1$	$\mathsf{s}(0)$		

where $\leftrightarrow_i$ stands for an application of equation i in either direction. There are two major observations in the above proof: First, the equations of the axiomatization were applied only from left to right, just directed steps $\rightarrow_1$ and $\rightarrow_2$ would have been sufficient. We call a directed equation $s \approx t$ a *rule* (written $s \to t$), sets of rules a *Term Rewriting System (TRS)*, and applications of rules to terms *rewriting*. In the context of TRSs, we will write i to denote orientation from left to right, and i' for right to left.

The second major observation is that no semantic insight was needed: Instead of deriving a proof, we can *compute* it. Thus, given a set of equations $\mathcal{E}$, instead of employing equational steps $\leftrightarrow_\mathcal{E}$, the goal is to orient the equations to a TRS $\mathcal{R}$ and employ only $\rightarrow_\mathcal{R}$. In the above example, the TRS $\{1,2\}$ suffices to efficiently construct a proof for any valid equation. This is because it has three important properties:

1. It is *terminating*: There exists no term s with infinite sequence $s \to s_1 \to \dots$
2. It is *confluent*: For all terms t, s, u such that $t \,{}^*\!\leftarrow s \to^* u$, there exists a term v with $t \to^* v \,{}^*\!\leftarrow u$. Thus, the result of a computation does not depend on the order of applications of rewrite rules.
3. The TRS is *equivalent* to the equational system, i.e. $\leftrightarrow_\mathcal{E}^* = \leftrightarrow_\mathcal{R}^*$. Thus two terms are reachable by equational steps if and only if they are convertible by rewriting.

Termination implies that for a given term, after finitely many steps, we reach a term where no rule is applicable — the given term's *normal form*. Confluence implies that this normal form is unique. Equivalence ensures that reachability by equational steps and reachability by directional steps coincide. We call a TRS *complete* for an equational system, if it is terminating, confluent and equivalent to the ES.

Unfortunately, not always can equational systems be turned into complete TRSs by just merely orienting its equations to rules. Consider an axiomatization of group-theory $\mathcal{E}_G$, consisting of

3:	$1 * x \approx x$	**neutral element**
4:	$x^{-1} * x \approx 1$	**inverse element**
5:	$(x * y) * z \approx x * (y * z)$	**associativity**

and the proof of $(x * 1) * x^{-1} \approx 1$. We have

$$\begin{aligned}
(x * 1) * x^{-1} &\leftrightarrow_5 x * (1 * x^{-1}) \\
&\leftrightarrow_3 x * x^{-1} \\
&\leftrightarrow_3 1 * (x * x^{-1}) \\
&\leftrightarrow_4 ((x^{-1})^{-1} * x^{-1}) * (x * x^{-1}) \\
&\leftrightarrow_5 (x^{-1})^{-1} * (x^{-1} * (x * x^{-1}) \\
&\leftrightarrow_5 (x^{-1})^{-1} * (x^{-1} * x) * x^{-1}) \\
&\leftrightarrow_4 (x^{-1})^{-1} * (1 * x^{-1}) \\
&\leftrightarrow_3 (x^{-1})^{-1} * x^{-1} \qquad\qquad \leftrightarrow_4 1
\end{aligned}$$

Using the TRS $\{3, 4, 5\}$ misses the proof, since then both $(x * 1) * x^{-1}$ and 1 are in normal form. The reason is that $\{3, 4, 5\}$ is terminating, but neither confluent, nor equivalent to the ES. The difficulty in finding a complete TRS is, that naively trying to fix one of the three properties easily leads to violation of another. For example $\{3, 4\}$ is both terminating and confluent, but not equivalent to the ES. $\{3, 3', 4, 4', 5, 5'\}$ is confluent and equivalent to the ES, but clearly not terminating.

Completion

Given an ES, Knuth and Bendix' [49] ground-breaking completion procedure systematically searches for an equivalent and complete TRS by orienting equations to rules, while maintaining confluence by resolving local peaks. For example, their procedure can construct the following complete TRS $\mathcal{R}_G$ for group-theory:

1:	$1 * x \to x$	6:	$1^{-1} \to 1$
2:	$x^{-1} * x \to 1$	7:	$(x^{-1})^{-1} \to x$
3:	$(x * y) * z \to x * (y * z)$	8:	$x * x^{-1} \to 1$
4:	$x^{-1} * (x * y) \to y$	9:	$x * (x^{-1} * y) \to y$
5:	$x * 1 \to x$	10:	$(x * y)^{-1} \to y^{-1} * x^{-1}$

The proof of $(x * 1) * x^{-1} \approx x$ is trivial with the complete TRS, since

$$(x * 1) * x^{-1} \to_5 x * x^{-1} \to_8 1 = 1.$$

So far we have only considered computing proofs for valid conjectures. To disprove a conjecture $s \approx t$ for a given ES $\mathcal{E}$, one has to verify the absence of equational steps to connect s and t, i.e. to show that $s \leftrightarrow_{\mathcal{E}}^* t$ is impossible. Since for each side of a conjecture potentially infinitely many derivations exist, it is usually not possible to exhaustively enumerate them.

One way to disprove a conjecture is to exploit the tight connection of *equational steps* and *logical entailment*. A conjecture is a logical consequence of an ES $\mathcal{E}$, if and only if all algebraic models of $\mathcal{E}$ are also a model of the conjecture. Due to Birkhoff's theorem [14], both notions of logical entailment and connectedness by equational steps are equivalent: A conjecture $s \approx t$ is a logical consequence of a given ES $\mathcal{E}$, if and only if $s \leftrightarrow_{\mathcal{E}}^{*} t$ holds. Thus one way to disprove a conjecture is to find a (counter-)model, and apply Birkhoff's theorem:

Consider again the equations of the ES $\mathcal{E}_G$, and take the model $\mathcal{M}$ with carrier set $\mathrm{GL}_2(\mathbb{R})$ of invertible 2×2-matrices over the field of real numbers

$$\mathrm{GL}_2(\mathbb{R}) = \left\{ \begin{bmatrix} a & b \\ c & d \end{bmatrix} \;\middle|\; a, \ldots, d \in \mathbb{R} \text{ and } ad - bc \neq 0 \right\}$$

with the standard matrix operations as interpretations of the function symbols, namely:

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} *_{\mathcal{M}} \begin{bmatrix} e & f \\ g & h \end{bmatrix} = \begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix},$$

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1_{\mathcal{M}}} = \frac{1}{ad - bc} *_{\mathcal{M}} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}, \text{ and}$$

$$1_{\mathcal{M}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Then $\mathcal{M}$ is a valid model of $\mathcal{E}_G$, since for all $A, B, C \in \mathrm{GL}_2(\mathbb{R})$ the three equations $A *_{\mathcal{M}} 1_{\mathcal{M}} = A$, $A^{-1_{\mathcal{M}}} *_{\mathcal{M}} A = 1_{\mathcal{M}}$, and $(A *_{\mathcal{M}} B) *_{\mathcal{M}} C = A *_{\mathcal{M}} (B *_{\mathcal{M}} C)$ hold.

The conjecture $x * y \approx y * x$ is not true, since not every group is abelian. In particular, $\mathcal{M}$ is a counter-model, because

$$\begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} *_{\mathcal{M}} \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} \neq \begin{bmatrix} 0 & 0 \\ 1 & 0 \end{bmatrix} *_{\mathcal{M}} \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}.$$

While simpler counter-models for group-theory exist, the construction of a counter-model is in general difficult to automate, as it requires semantic insight into the ES — most Lie-groups are non-commutative.

On the other hand one can also make use of the following lemma, combining Birkhoff's theorem with completion: Given a complete TRS $\mathcal{R}$ for an ES $\mathcal{E}$, a conjecture $s \approx t$ is a logical consequence of $\mathcal{E}$, if and only if the normal forms of s and t w.r.t. $\mathcal{R}$ are syntactically equal. This very much simplifies the disproof of $x * y \approx y * x$: Both $x * y$ and $y * x$ are in normal form w.r.t. $\mathcal{R}_G$, and not syntactically equal. Therefore we can immediately reject the conjecture — without the need to explicitly construct a counter-model.

Note how all three properties of $\mathcal{R}_G$ are essential: Confluence implies the *Church-Rosser property* $\leftrightarrow_{\mathcal{R}_G}^{*} = \rightarrow_{\mathcal{R}_G}^{*} \cdot {}_{\mathcal{R}_G}^{*}\leftarrow$, by equivalence $\leftrightarrow_{\mathcal{E}_G}^{*} = \leftrightarrow_{\mathcal{R}_G}^{*}$ holds,

and termination ensures the existence of normal forms and together with confluence their uniqueness. Finally $\leftrightarrow^*_{\mathcal{E}_G}$ connects to logical entailment by Birkhoff's theorem.

Challenges in Completion

Deciding validity of equations is undecidable in general. Thus it is not surprising that we cannot always find a complete TRS for a given ES. But even if a complete TRS exists, it is no trivial task to find it. Knuth and Bendix' completion procedure poses several challenges:

1. The procedure is formulated as an iterative algorithm, but its formulation includes several *indeterminism*, which have to be resolved in an effective way to achieve full automation.

2. The procedure assumes a *reduction order* as an input parameter to ensure termination. It is in general a highly non-trivial task to find a suitable reduction ordering a priori.

3. Its *algorithmic formulation* makes it very difficult to distinguish essential properties from implementational details. This is especially a hindrance, when trying to extend the procedure to resolve the above two issues.

Inductive Reasoning

Depending on the area of application, the notion of logical entailment and connectedness via equational steps does not capture the properties one is interested in. Consider the following axiomatization $\mathcal{E}_@$ of list-append, that almost verbatim corresponds to code in functional programming languages:

$$\begin{aligned} 1\colon &\quad [\,] \mathbin{@} xs \approx xs \\ 2\colon &\quad (x :: xs) \mathbin{@} ys \approx x :: (xs \mathbin{@} ys) \end{aligned}$$

The ES $\mathcal{E}_@$ does not logically entail the conjecture

$$3\colon \quad xs \mathbin{@} [\,] \approx xs.$$

Take the model $\mathcal{M}$ with carrier set $\{0,1\}$, and interpretations $[\,]_\mathcal{M} = 0$, $x ::_\mathcal{M} xs = xs$, and $xs \mathbin{@}_\mathcal{M} ys = ys$. One can verify, that $\mathcal{M}$ is indeed a model of $\mathcal{E}_@$. But $\mathcal{M}$ is a counter-model of the conjecture, since $1 \mathbin{@}_\mathcal{M} 0 = 0 \neq 1$.

When executing a program however, one can be sure that the data structures operated on are lists, and thus any instance of the variables is constructed by using the operator :: and the empty list $[\,]$. Taking this into account, one can prove the conjecture $xs \mathbin{@} [\,] \approx xs$ by *induction* on these constructor symbols.

For the proof, we distinguish two cases: If xs is instantiated as $[\,]$, then trivially $[\,] \mathbin{@} [\,] = [\,]$. On the other hand, if xs has the form $x :: xs'$ where x and xs' are free, we have

	lhs	**rhs**
	$(x :: xs') \mathbin{@} [\,]$	$x :: xs'$
$\leftrightarrow_2$	$x :: (xs' \mathbin{@} [\,])$	
$\leftrightarrow_3$	$x :: xs'$	

Thus the conjecture does not hold in general, but it does hold *on all ground instances*. Observations on the computation of this proof are similar to the introductory example of addition: Again, we used both equations from $\mathcal{E}_@$ and the hypothesis only from left to right, we could have conducted the proof using the TRS $\mathcal{R}_@ = \{1, 2, 3\}$ instead. The main difference and difficulty compared to completion is, that extra care is needed to ensure that the application of the hypothesis is sound. For example, both rule 1 and rule 3 are applicable on the left-hand side of $(x :: xs') \mathbin{@} [\,] \approx x :: xs'$. Using rule 3 however leads to unsound reasoning, since it means self-application of the hypothesis. Another difference is, that due to its application, we are rarely interested in disproving but rather in *proving* conjectures. To compute a proof, one needs to ensure that all ground instances of both sides of the conjecture are joinable by rewrite steps of the TRS, and by sound application of rewrite steps using the hypothesis. Confluence of the TRS together with the hypothesis can be a sufficient criterion to ensure that, but it is not a necessary one, and in practice usually too strong.

Challenges in Inductive Reasoning

Inductive reasoning is considered to be even more challenging to automate than standard equational reasoning. For a more in-depth treatment, we refer to [36], but mention the following issues:

1. The proof *strategy*. Proofs usually have a recursive structure, yet it is not clear when and how to soundly apply the hypothesis. Since confluence, and even termination can be too strong properties to assume, it is also not clear which rewrite strategy should be applied.

2. The need of *lemmata*. Very often, direct proof attempts fail, and must be split up into several sub-goals, for which additional lemmata are needed. Since this usually requires semantic insight, it is very challenging to identify such sub-goals and generate appropriate lemmata automatically.

3. The lack of an expressive *framework*. Different inductive proof techniques exist. But identifying their essential properties is difficult, as soundness is

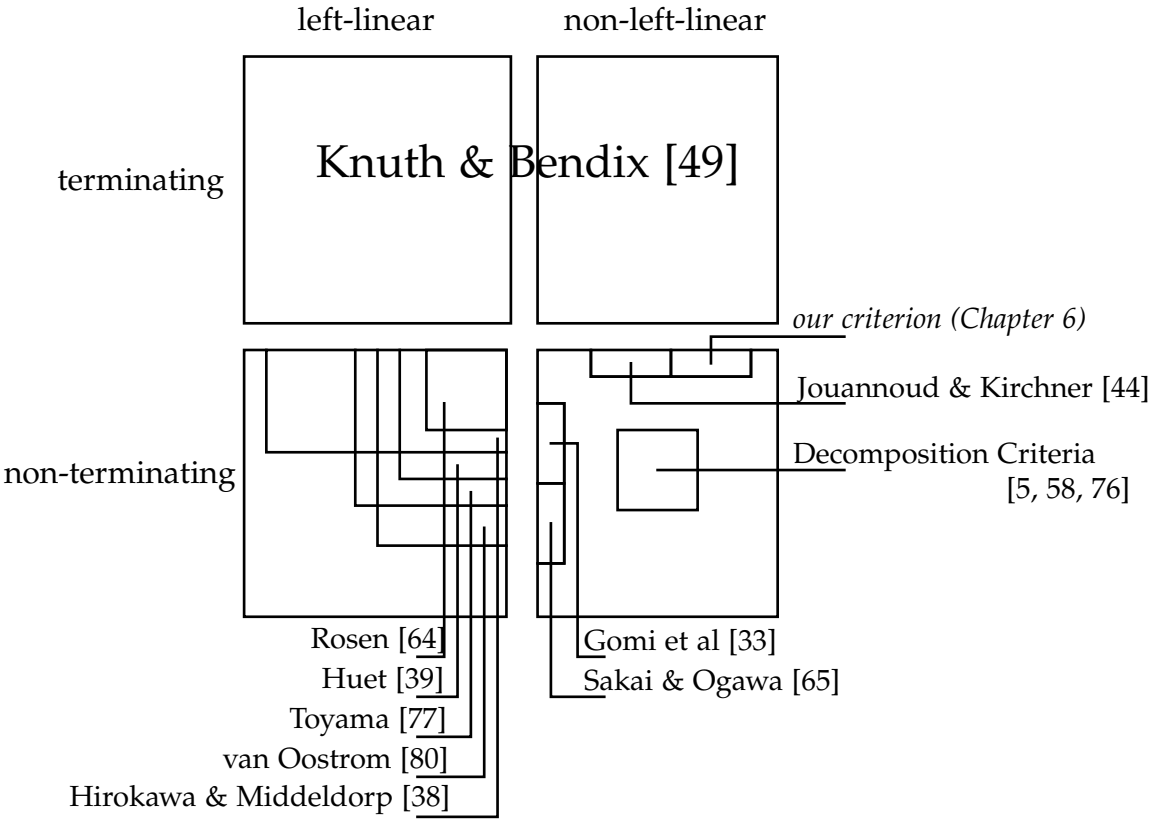

Figure 1.1: A non-exhaustive overview of confluence criteria.

established by ensuring abstract joinability requirements on ground terms, which are hard to express and reason about.

Confluence

Confluence is an important property in various domains. For TRSs, confluence is equivalent to the Church-Rosser property $\leftrightarrow^* \subseteq \rightarrow^* \cdot {}^*\leftarrow$. Church and Rosser [16] investigated this property to ensure *consistency* of combinatory logic and the λ-calculus. Both form the foundation for the development of functional programming languages such as SML [55], OCaml[1] or Haskell [61]. Other areas of application include narrowing [24, 42] and type safety [73].

From the perspective of equational reasoning, confluence together with termination ensures completeness. Both conditions impose severe restrictions on the TRS, and naturally one wants to relax both of them. Relaxing the confluence condition is however almost impossible, since it is tied so closely to the concept of orienting equations to rules. On the other hand confluence implies $\leftrightarrow^*_{\mathcal{R}} \subseteq \rightarrow^*_{\mathcal{R}} \cdot {}^*_{\mathcal{R}}\leftarrow$. Even if termination does not hold, this still suggests a simple

[1] `http://caml.inria.fr`

semi-decision procedure by enumerating $\rightarrow_{\mathcal{R}}^{n} \cdot {}_{\mathcal{R}}^{m}\leftarrow$ with increasing n and m. Suitable *strategies* to apply rewrite rules can help to reduce the size of the search tree significantly. Thus it is more promising to relax the termination condition, and identify criteria for confluence that do not rely on it.

Challenges to Develop Confluence Criteria

Knuth and Bendix [49] showed that confluence of terminating TRSs is decidable by testing joinability of so-called critical pairs. For non-terminating TRSs, very often syntactical criteria are employed, in particular *left-linearity*, i.e. each variable in a left-hand side of a rule may occur only once. A famous result due to Rosen [64] is that a left-linear TRS without critical pairs is confluent. However the fact that the complete TRS $\mathcal{R}_G$ for group-theory includes non-left-linear rules already gives a hint that this is a too strong property to presuppose in practice. We list as challenges:

1. Proving confluence of both *non-left-linear and non-terminating* TRSs remains extremely difficult, and only few results exist. One reason is that none of the notions and techniques, for example *critical pairs*, that are used in showing confluence of terminating or left-linear TRSs are suitable in the non-left-linear and non-terminating case.

2. Extending these notions can lead to new criteria, but often such extensions face the problem of *computability*. For example, extending the notion of critical pairs quickly leads to infinite sets of equations that are uncomputable in general. Thus, despite developing criteria, *automation* is hard to obtain.

1.2 Overview and Contributions

This thesis is structured as follows: *Chapter 2* is an introduction to term rewriting systems. First, abstract reduction systems are introduced. Properties such as confluence and termination can already be defined on this abstract level, and will be used when later specializing the notion of abstract reduction systems to term rewriting systems. Also formally introduced are several kinds of equational problems that will be treated in the later chapters, namely equational reasoning and inductive reasoning.

Chapter 3 recalls existing completion procedures. The original Knuth-Bendix completion procedure and Huet's algorithm [40] are introduced, as well as its reformulation by means of inference rules [12]. We also recollect approaches that increase the power of the original algorithm by automatically finding a suitable reduction ordering, such as Multi-Completion [51], Completion with Termination Tools [82], and Multi-Completion with Termination Tools [68, 84].

During the early 2000's, significant progress has been made in automated constraint solving. In a lot of areas, encoding problems as boolean constraints and solving them with SAT solvers has turned out to be more effective than the development of dedicated algorithms, and have largely replaced traditional approaches. Here we mention bounded model checking [13] and termination analysis of TRSs [17, 26, 70, 87]. To obtain efficient encodings to SAT, the search space of the underlying problem has to be known, and representable by a finite number of constraints. Unfortunately, in the case of completion, the search space that includes all complete TRSs is usually unknown in advance. Therefore it is non-trivial to directly encode the completion problem as a satisfiability problem.

Inspired by the progress in satisfiability solving, also several tools for MaxSAT solving appeared. Compared to SAT solving however, few of these systems have seen application outside their very domain. In *Chapter 4*, we present *maximal completion*. We characterize completion as an optimization problem, where the goal is to find an optimal TRS among a set of (exponentially) many. This task can be encoded as a maximal satisfiability problem and automatically solved by a MaxSAT (or MaxSMT) solver. We relate maximal completion with existing completion procedures, and compare our implementation `Maxcomp` with state-of-the-art completion tools.

Maximality is the key ingredient in maximal completion. To apply maximality in adaptations of completion for equational reasoning, we propose a framework for equational reasoning based on *constrained equalities* in *Chapter 5*. This framework is universal enough to formulate various joinability requirements for equational reasoning tasks, including the more complex requirements of methods for inductive reasoning, such as inductionless induction and rewriting induction. Most importantly it enables us to reformulate the various joinability conditions as optimization problems and thus exploit maximality. Experiments for inductive proof methods such as rewriting induction show practical effectiveness. Additionally the framework also allows to recover inter-reduction for completion that was lost in the abstraction of maximal completion. Experimental evaluation however shows that for completion the approach used in maximal completion is more efficient.

For non-terminating TRSs, several criteria exist to establish confluence. Figure 1.1 gives an non-exhaustive overview of confluence criteria on four classes of TRSs related to our setting. Note here that criteria for non-terminating TRSs can of course also be applied to terminating ones. Likewise those for non-left-linear TRSs apply to left-linear ones as well. As mentioned, for terminating TRSs confluence is decidable by testing joinability of critical pairs, which are induced by *overlaps*. Criteria for the case of non-terminating and left-linear TRSs usually build up on the orthogonality result by Rosen [64], either by a more precise analysis of overlaps [35, 38, 39, 77, 80] or by employing *relative termination*. Approaches

for the case of non-termination and non-confluence can be roughly classified into three categories:

1. By generalizing the notion of overlaps, one can formulate *direct criteria* [31, 65] that ensure confluence. Usually, strong syntactic restrictions are imposed to handle the very complex overlap analysis.

2. By *decomposing* the TRS [5, 25, 58, 59, 76] into smaller ones, one can employ existing criteria to show confluence for each of them. *Modularity* of the decomposition then ensures that the union remains confluent.

3. Criteria that relax termination requirements to relative termination. To apply equational reasoning for theories where no complete TRSs exists, Jouannaud and Kirchner gave a confluence criterion for the Church-Rosser modulo property [44] based on so called extended critical pairs. Later Geser [27] analyzed their proof and gave a criterion based on purely syntactic critical pairs.

In Chapter 6 we first recall some definitions and techniques, such as extended critical pairs, to build our tool-set for later proofs. Then we introduce our confluence criterion for non-left-linear and non-terminating TRSs, which also relies on relative termination. Similar to Jouannaud and Kirchner's criterion, it requires to check joinability of (uncomputable) extended critical pairs, but we show that for certain classes of TRSs, joinability of (computable) syntactic critical pairs suffices. We then compare our criterion with existing results, and provide experimental data.

Contributions

As main contributions of this thesis we list:

1. *Maximal completion*. We show that completion can be characterized as an optimization problem. In contrast to existing procedures, our reformulation does not contain any algorithmic aspect. It abstracts away from implementational details by encoding the completion requirements to constraints. It is less reliant on heuristics to resolve indeterminism than existing procedures, and suitable reduction orderings are constructed automatically based on solutions of the constraint problem.

2. *Constrained Equalities*. This framework allows not only to uniformly express the joinability requirements of completion and several procedures for inductive reasoning, but more importantly also makes it possible to reformulate these procedures as optimization problems, similar to maximal completion. While for completion itself practically not as efficient as

maximal completion, it provides a powerful technique to implement fully automated inductive theorem provers. As a theoretical interest, it recovers inter-reduction in completion, an optimization technique that is lost in the abstraction used in maximal completion.

3. A new *confluence criterion* for *non-left-linear* and *non-terminating* TRSs. Our criterion is based on moderate syntactic restrictions and relative termination, and requires joinability of extended critical pairs. In order to overcome the uncomputability of extended critical pairs, we provide a new *result on equational unification*, and with it, our confluence criterion can be fully automated.

4. All of the above approaches have been implemented and experimentally evaluated to show their effectiveness. Moreover, we provide the free completion tool MAXCOMP, which is based on maximal completion.

All results are included in the papers [4, 46, 47].

Chapter 2

Preliminaries

In this chapter we formally introduce term-rewriting, and the equational problems that we will tackle. We restrict us to the essential definitions and properties needed for the later chapters. For full proofs and a more gentle, comprehensive introduction to term-rewriting we refer to [10, 60, 75], and [9] (the latter in German).

2.1 Abstract Rewriting

A (binary) relation $\to$ on a set A is a subset of $A \times A$. Given two relations $\to_1$ and $\to_2$ on A, the *composition* $\to_1 \cdot \to_2$ is defined as

$$\{(x, z) \in A \mid \text{there exists } y \in A \text{ such that } x \to_1 y \text{ and } y \to_2 z\}.$$

Definition 2.1.1. *An* Abstract Rewrite System *(ARS)* $\mathcal{A} = (A, \langle \to_\alpha \rangle_{\alpha \in I})$ *is a tuple consisting of a set* A *and a set of relations* $\to_\alpha$ *on* A*, where* I *is a set of* labels*. We say* $\to_\alpha$ *is a reduction relation* labelled *with* α*. The reduction relation* $\to_{\mathcal{A}}$ of $\mathcal{A}$ *is defined as*

$$\to_{\mathcal{A}} = \bigcup_{\alpha \in I} \to_\alpha.$$

We sometimes write $\to$ instead of $\to_{\mathcal{A}}$, if $\mathcal{A}$ is clear from the context. In the following definitions, we assume I to be a singleton and simply write $\mathcal{A} = (A, \to)$. One motivation to introduce labels is that it is sometimes useful to see $(A, \langle \to_\alpha \rangle_{\alpha \in I})$ as a representation of $(A, \to)$. In particular the *decreasing diagram* technique [79] used in Chapter 6 will make use of this.

Definition 2.1.2. *Let $\mathcal{A} = (A, \to)$ be an ARS. We define the following notions:*

$\to^0$	$=$	$\{(a,a) \mid a \in A\}$	identity
$\to^{i+1}$	$=$	$\to^i \cdot \to$	$(i+1)$-fold composition
$\to^+$	$=$	$\bigcup_{i>0} \to^i$	transitive closure
$\to^=$	$=$	$\to^0 \cup \to$	reflexive closure
$\to^*$	$=$	$\to^0 \cup \to^+$	transitive reflexive closure
$\leftarrow$	$=$	$\{(b,a) \mid (a,b) \in \to\}$	inverse relation
$\leftrightarrow$	$=$	$\to \cup \leftarrow$	symmetric closure
$\leftrightarrow^+$	$=$	$(\leftrightarrow)^+$	transitive symmetric closure
$\leftrightarrow^*$	$=$	$(\leftrightarrow)^*$	reflexive transitive symmetric closure
$\downarrow$	$=$	$\to^* \cdot {}^*\leftarrow$	join relation

We will make use of the following terminology:

- We will write $a \to b$ if $(a,b) \in \to$, and say that *a rewrites to b* and *b is a reduct of a*.
- We say that a and b are *joinable*, if $a \downarrow b$.

We now define the central notions of *confluence* and *termination* for ARSs and investigate their relation with normal forms.

Definition 2.1.3. *Let $\mathcal{A} = (A, \to)$ be an ARS.*

1. *$a \in A$ is in* normal form *(w.r.t. $\to$), if there exists no b such that $a \to b$.*
2. *b is* a normal form of *a, if $a \to^* b$, and b is in normal form. If a has a unique normal form b, we write $a{\downarrow_{\mathcal{A}}}$ for b. In some contexts, we will ambiguously write $a{\downarrow_{\mathcal{A}}}$ to denote a specific normal form. The set of normal forms of A is written $\mathsf{NF}(A)$ or $\mathsf{NF}(\to)$, if A is clear from the context.*
3. *$\mathcal{A}$ is* normalizing, *if for every $a \in A$ there exists a $b \in A$ such that b is a normal from of a.*
4. *$\mathcal{A}$ is* uniquely normalizing, *if for every $a \in A$ there exists exactly one $b \in A$ such that b is a normal from of a.*

5. $\mathcal{A}$ *is* terminating, *if there exists no infinite sequence* $a_0 \to a_1 \to a_2 \to a_3 \to \dots$

Definition 2.1.4. *Let* $\mathcal{A} = (A, \to)$ *be an ARS.*

1. $\mathcal{A}$ *is* Church-Rosser *if* $\leftrightarrow^* \subseteq \downarrow$.
2. $\mathcal{A}$ *is* confluent *if* ${}^*\!\leftarrow \cdot \to^* \subseteq \downarrow$.
3. $\mathcal{A}$ *is* locally confluent *(or* weakly Church-Rosser*) if* $\leftarrow \cdot \to \subseteq \downarrow$.

Lemma 2.1.5. *A relation* $\to$ *is confluent if and only if it is Church-Rosser.* ∎

Termination localizes the confluence condition.

Lemma 2.1.6 (Newman [56])**.** *A terminating ARS* $\mathcal{A}$ *is confluent if and only if it is locally confluent.* ∎

Definition 2.1.7. *An ARS* $\mathcal{A}$ *is* complete, *if it is confluent and terminating.*

Confluence and unique normal forms are connected:

Lemma 2.1.8. *Let* $\mathcal{A}$ *be an ARS.*

- *Suppose* $\mathcal{A}$ *is confluent. Then each* $a \in \mathcal{A}$ *has at most one normal form.*
- *Suppose* $\mathcal{A}$ *is complete. Then* $\mathcal{A}$ *is uniquely normalizing.*

∎

Completeness of an ARS $\mathcal{A}$ provides a way to test $a \leftrightarrow^*_{\mathcal{A}} b$.

Theorem 2.1.9. *Let* $\mathcal{A}$ *be a complete ARS. Then* $a \leftrightarrow^*_{\mathcal{A}} b$ *if and only if* $a{\downarrow_{\mathcal{A}}} = b{\downarrow_{\mathcal{A}}}$. ∎

2.2 Orders

A binary relation $\to$ on A is *well-founded*, if there exists no infinite sequence $a_0 \to a_1 \to a_2 \to \cdots$ of elements. One might wonder, why we call it well-founded and not, like ARSs $(A, \to)$ without infinite sequence $a_0 \to a_1 \to a_2 \to \dots$, terminating. It is convenient to use a separate terminology here, since we later specialize ARSs to term rewriting systems (TRSs), and employ certain kinds of well-founded orders to automatically show termination of TRSs.

Definition 2.2.1. *A binary relation on* A *is*

1. *a* strict order, *if it is irreflexive and transitive*
2. *a* preorder, *if it is reflexive and transitive and*

3. *a* partial order, *if it is reflexive, transitive and anti-symmetric.*

Definition 2.2.2. *$A >$ strict order on A is*

1. *a* total order, *if for all $a, b \in A$ we have $a > b$, $b > a$ or $a = b$.*
2. *a* well-founded order, *if $>$ is well-founded and*
3. *a* well-order, *if it is both total and well-founded.*

We introduce two extensions of strict orders. The first is the lexicographic extension.

Definition 2.2.3. *Let $>$ be a strict order on A. The* lexicographic extension $>^{lex}$ *on A^n is defined as: $(s_1, \ldots, s_n) >^{lex} (t_1, \ldots, t_n)$ if and only if there exists some i with $0 \leq i < n$ such that $s_j = t_j$ for all $1 \leq j \leq i$ and $s_{i+1} > t_{i+1}$.*

The lexicographic extension preserves well-foundedness.

Theorem 2.2.4 ([10])**.** *Let $>$ be a strict ordering. The lexicographic extension $>^{lex}$ is well-founded if $>$ is well-founded.* ∎

The second extension is based on multisets.

Definition 2.2.5. *Let A be a set. A* multiset *M over A is a function $M : A \to \mathbb{N}$. A multiset M is* finite*, if there are only finitely many x such that $M(x) > 0$. The set of all finite multisets over A is denoted by $\mathcal{M}(A)$. Let $a_1, \ldots, a_n$ be elements of A. We write $\{\!\{a_1, \ldots, a_n\}\!\}$ for the multiset M defined as follows: $M(a)$ denotes the size of the set $\{i \mid 1 \leq i \leq n, a_i = a\}$. Let M and N be multisets. The following operations are defined:*

- *$x \in M$, if $M(x) > 0$*
- *$M \subseteq N$, if $M(x) \leq N(x)$ for all $x \in A$.*
- *$M \cup N$ stands for $x \mapsto M(x) + N(x)$, and*
- *$M \setminus N$ stands for $x \mapsto \max\{0, M(x) - N(x)\}$.*

Definition 2.2.6. *Let $>$ be an order on A. The* multiset extension $>^{mul}$ *of $>$ is defined on $\mathcal{M}(A)$ as:*

$M >^{mul} N$ if and only if there exists $X, Y \in \mathcal{M}(A)$ such that
a) $\{\!\{\ \}\!\} \neq X \subseteq M$,
b) $N = (M \setminus X) \cup Y$, and
c) for every $y \in Y$ there exists an $x \in X$ with $x > y$

Theorem 2.2.7 ([10])**.** *Let $>$ be a strict ordering. The multiset extension $>^{mul}$ is well-founded if and only if $>$ is well-founded.* ∎

2.3 Terms and Substitutions

Definition 2.3.1. *A* signature $\mathcal{F}$ *is a countable set of* function symbols*, where each* $f \in \mathcal{F}$ *is associated with a natural number denoting its* arity *(the number of arguments). We write* $f^{(n)}$ *when we want to explicitly express that the arity of* f *is* n*. A function symbol of arity zero is called a* constant*.*

Definition 2.3.2. *Let* $\mathcal{F}$ *be a signature and* $\mathcal{V}$ *be a countable set of* variables *with* $\mathcal{F} \cap \mathcal{V} = \varnothing$*. The set of* terms $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *over* $\mathcal{F}$ *and* $\mathcal{V}$ *is the smallest set such that*

- $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$*, and*
- *if* $f^{(n)} \in \mathcal{F}$ *and* $t_1, \ldots, t_n \in \mathcal{T}(\mathcal{F}, \mathcal{V})$*, then* $f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$*.*

When denoting specific function symbols $\mathsf{f}, \mathsf{g}, \ldots$ we will use a gothic (sans-serif) font, and for variables $x, y, \ldots$ we will use *italics*.

Definition 2.3.3. *Let* $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *be a term.* $\mathcal{V}\mathsf{ar}(t)$ *denotes the* set of variables *occurring in* t*:*

$$\mathcal{V}\mathsf{ar}(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ \bigcup_{i=1}^{n} \mathcal{V}\mathsf{ar}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Similarly, $\mathcal{F}\mathsf{un}(t)$ *denotes the* set of function symbols *(including constants) of* t*:*

$$\mathcal{F}\mathsf{un}(t) = \begin{cases} \varnothing & \text{if } t \text{ is a variable} \\ \{f\} \cup \bigcup_{i=1}^{n} \mathcal{F}\mathsf{un}(t_i) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Definition 2.3.4. *For* $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *the* set of positions $\mathcal{P}\mathsf{os}(t)$ *is defined as*

$$\mathcal{P}\mathsf{os}(t) = \begin{cases} \{\varepsilon\} & \text{if } t \text{ is a variable} \\ \{\varepsilon\} \cup \{i.q \mid 1 \leq i \leq n, q \in \mathcal{P}\mathsf{os}(t_i)\} & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Here ε *denotes the empty string. A position is thus a sequence of integers separated by the* . *symbol. The position* ε *is the* root position*. The* root symbol *of* t *is defined as:*

$$\mathsf{root}(t) = \begin{cases} t & \text{if } t \text{ is a variable} \\ f & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

Finally $\mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ *is defined as the set* $\{p \in \mathcal{P}\mathsf{os}(t) \mid \mathsf{root}(t|_p) \in \mathcal{F}\}$*.*

A position describes the path from the root to a specific subterm.

Definition 2.3.5. *Let t be a term and $p \in \mathcal{P}\mathsf{os}(t)$. The* subterm $t|_p$ *of t at position p is defined as:*

$$t|_p = \begin{cases} t & \text{if } p = \varepsilon \\ t_i|_q & \text{if } p = i.q \text{ and } t = f(t_1, \ldots, t_i, \ldots, t_n) \end{cases}$$

With $t[u]_p$ we denote the operation of replacing the subterm $t|_p$ of t at p with the term u:

$$t[u]_p = \begin{cases} u & \text{if } p = \varepsilon \\ f(t_1, \ldots, t_i[u]_q, \ldots, t_n) & \text{if } p = i.q \text{ and } t = f(t_1, \ldots, t_n) \end{cases}$$

A term u is called a subterm *of a term t, if there exists a position $p \in \mathcal{P}\mathsf{os}(t)$, such that $u = t|_p$. Given positions p, q, and o, we write $p \backslash q$ for o if $p = q.o$. Let x be a variable. We will write $|t|_x$ for $\left|\{p \mid p \in \mathcal{P}\mathsf{os}(t) \text{ and } t|_p = x\}\right|$, i.e. the number of occurrences of x in t.*

Definition 2.3.6. *A term t is*

- *a* ground term *if $\mathcal{V}\mathsf{ar}(t) = \varnothing$. Instead of $\mathcal{T}(\mathcal{F}, \varnothing)$ we sometimes write $\mathcal{T}(\mathcal{F})$ to denote the set of all ground terms over the signature $\mathcal{F}$, and*
- linear *if each variable appears only once, i.e. if for all $x \in \mathcal{V}\mathsf{ar}(t)$, we have $|t|_x = 1$.*

The *size* $|t|$ of a term is defined as $|\mathcal{P}\mathsf{os}(t)|$, that is the number of functions symbols and variables occurring in it.

Definition 2.3.7. *Let $\mathcal{F}$ be a signature, and let $\Box$ denote a special constant symbol, called the* hole. *A* context *over the union of the signature and the hole $\mathcal{F} \cup \{\Box\}$ is a term C, such that C contains exactly one hole, i.e. $\left|\{p \in \mathcal{P}\mathsf{os}(t) \mid t|_p = \Box\}\right| = 1$. The* application $C[t]$ *of a context C on a term t is defined as:*

$$C[t] = \begin{cases} t & \text{if } C = \Box \\ f(t_1, \ldots, C'[t], \ldots, t_n) & \text{if } C = f(t_1, \ldots, C', \ldots, t_n) \end{cases}$$

Here C' denotes a context.

Definition 2.3.8. *Let $\mathcal{F}$ be a signature, and $\mathcal{V}$ be a countable set of variables, such that $\mathcal{F} \cap \mathcal{V} = \varnothing$. A* substitution *$\sigma$ over $\mathcal{T}(\mathcal{F}, \mathcal{V})$ is a function $\mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$, such that the* domain *$Dom(\sigma)$*

$$Dom(\sigma) = \{x \in \mathcal{V} \mid \sigma(x) \neq x\}$$

is finite. When referring to a specific substitution σ, we sometimes write it as

$$\{x_1 \mapsto t_1, x_2 \mapsto t_2, \ldots, x_n \mapsto t_n\}$$

using the finiteness of the domain. The application of a substitution σ on a term t, denoted $t\sigma$, is defined as follows:

$$t\sigma = \begin{cases} \sigma(t) & \text{if } t \in \mathcal{V} \\ f(t_1\sigma, \ldots, t_n\sigma) & \text{if } t = f(t_1, \ldots, t_n) \end{cases}$$

We say $t\sigma$ is an instance *of t. The* composition *of two substitutions σ and τ is defined as $(\sigma\tau)(x) = \sigma(\tau(x))$. We usually simply write $x\sigma\tau$. A substitution $\sigma : \mathcal{V} \to \mathcal{V}$ is called a* variable substitution*, and a bijective variable substitution is called a* variable renaming*. A substitution σ with $\sigma(x) \in \mathcal{T}(\mathcal{F}, \varnothing)$ for all $x \in \mathcal{V}\mathsf{ar}(t)$ is called a* ground substitution *on t. A substitution σ is a ground substitution on an equation $s \approx t$, if it is both a ground substitution on s and on t.*

Definition 2.3.9. *Let s and t be terms. We define the following notions:*

- *s and t are* syntactically unifiable *if there exists a substitution σ with $s\sigma = t\sigma$.*
- *If $s\sigma = t\sigma$ holds for a substitution σ, then we call σ a* unifier *of s and t.*
- *A unifier σ of s and t is a* most general unifier (mgu) *if for every unifier τ of s and t there exists some substitution ρ such that $\sigma\rho = \tau$.*

Syntactical unifiability is decidable ([10, 63]), and unifiable terms admit an mgu, which is unique up to variable renamings. We will extend the notion of unifiability to equational unifiability in Chapter 6, where these extensions and several of their properties will be used. When speaking of unifiability and unifiers, we usually mean syntactic unifiability and syntactic unifiers, unless noted otherwise.

2.4 Term Rewriting Systems

Definition 2.4.1. *Let $\to$ be a relation on terms. It is*

- monotonic (or closed under contexts) *if $C[s] \to C[t]$ holds for all s and t with $s \to t$ and all contexts C,*
- stable (or closed under substitutions) *if $s\sigma \to t\sigma$ holds for all s and t with $s \to t$ and all substitutions σ,*
- *a* rewrite relation *if it is monotonic and stable,*
- *a* rewrite order *if it is a strict order, monotonic and stable, and*
- *a* reduction order *if it is a well-founded rewrite order.*

Definition 2.4.2. *A* Term Rewriting System (TRS) *is a tuple* $(\mathcal{F}, \mathcal{R})$ *consisting of a signature* $\mathcal{F}$ *and a set* $\mathcal{R} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}, \mathcal{V})$ *of* rewrite rules. *A rewrite rule* (ℓ, r) *must satisfy*

- $\ell \notin \mathcal{V}$, *and*
- $\mathcal{V}\mathsf{ar}(\ell) \supseteq \mathcal{V}\mathsf{ar}(r)$.

We write $\ell \to r$ *instead of* (ℓ, r). *The* rewrite relation $\to_{\mathcal{R}}$ *of a TRS* $(\mathcal{F}, \mathcal{R})$ *is defined as:* $s \to_{\mathcal{R}} t$ *if there exists a position* $p \in \mathcal{P}\mathsf{os}(s)$, *a substitution* σ, *and a rule* $\ell \to r \in \mathcal{R}$ *such that* $s = s[\ell\sigma]_p$ *and* $t = s[r\sigma]_p$. *We say* $s \to_{\mathcal{R}} t$ *is a* rewrite step. *A rule* $\ell' \to r'$ *is a* variant of a rule $\ell \to r$, *if there exists a variable renaming* σ, *such that* $\ell\sigma = \ell'$ *and* $r\sigma = r'$.

By definition, $s \to_{\mathcal{R}} t$ implies the existence of a position $p \in \mathcal{P}\mathsf{os}(s)$, a rule $\ell \to r \in \mathcal{R}$ such that $s = s[\ell\sigma]_p$ and $t = s[r\sigma]_p$. When we want to make the position and/or rule explicit, we will write $s \xrightarrow{p}_{\ell\to r} t$, $s \xrightarrow{p}_{\mathcal{R}} t$ or $s \to_{\ell\to r} t$. We often omit $\mathcal{F}$ when referring to a TRS, simply write $\mathcal{R}$ and assume, unless otherwise noted, the signature of $\mathcal{R}$ to consist of all function symbols occurring in $\mathcal{R}$.

Definition 2.4.3. *Let* $\ell \to r$ *be a rule of the TRS* $\mathcal{R}$. *It is called*

- left-linear *if* ℓ *is linear,*
- right-linear *if* r *is linear,*
- linear *if both* ℓ *and* r *are linear,*
- collapsing, *if* $r \in \mathcal{V}$, *and*
- erasing *if not each variable that occurs in the left-hand side also occurs at least once in the right hand side.*

Note that all properties of ARSs, in particular confluence and termination trade over to TRS when treating a TRS $\mathcal{R}$ with signature $\mathcal{F}$ as an ARS over the set of terms and relation $\to_{\mathcal{R}}$, i.e. as the ARS $(\mathcal{T}(\mathcal{F}, \mathcal{V}), \to_{\mathcal{R}})$. In general, confluence and termination or TRSs are undecidable [41, 10]. But several sufficient criteria to detect confluence of a TRS exist, which are based on overlaps:

Definition 2.4.4. *Let* $\mathcal{R}_1$ *and* $\mathcal{R}_2$ *be TRSs. Then an* overlap of $\mathcal{R}_1$ on $\mathcal{R}_2$ *is a tuple* $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\sigma$ *consisting of a position* $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, *a substitution* σ, *and variants of rules* $\ell_1 \to r_1 \in \mathcal{R}_1$ *and* $\ell_2 \to r_2 \in \mathcal{R}_2$, *such that following conditions hold:*

- $\ell_2|_p \notin \mathcal{V}$,
- *if* $\ell_1 \to r_1$ *is a renaming of* $\ell_2 \to r_2$, *then* $p \neq \varepsilon$, *and*

- $\ell_2|_p$ *and* ℓ_1 *unify with mgu* σ.

This overlap induces the critical pair $\ell_2\sigma[r_1\sigma]_p \leftarrow \rtimes \rightarrow r_2\sigma$. *A critical pair is* trivial, *if* $\ell_2\sigma[r_1\sigma]_p = r_2\sigma$. *Given a TRSs* $\mathcal{R}_1$ *and* $\mathcal{R}_2$, *the set of critical pairs generated by all overlaps of* $\mathcal{R}_1$ *on* $\mathcal{R}_2$ *and all overlaps on* $\mathcal{R}_2$ *on* $\mathcal{R}_1$ *is written as* $\mathsf{CP}(\mathcal{R}_1, \mathcal{R}_2)$. *We write* $\mathsf{CP}(\mathcal{R})$ *for* $\mathsf{CP}(\mathcal{R}, \mathcal{R})$. *A TRS* $\mathcal{R}$ *is* non-overlapping *if* $\mathsf{CP}(\mathcal{R}) = \varnothing$. *A non-overlapping and left-linear TRS is called* orthogonal.

We remark that a more general notion of overlap and critical pair exists that subsumes the above definition. We will make use of this more general notion when introducing our confluence criterion in Chapter 6. Since we do consider these notions non-standard, we will not treat them here but define and discuss them when needed in Chapter 6.

The following result by Knuth and Bendix shows decidability of confluence for terminating TRSs.

Theorem 2.4.5 (Knuth and Bendix, [49])**.** *Let* $\mathcal{R}$ *be a terminating TRS. Then* $\mathcal{R}$ *is confluent if and only if all critical pairs are joinable, that is* $\mathsf{CP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$. ∎

For non-terminating TRSs, the most famous result is due to Rosen:

Theorem 2.4.6 (Rosen, [64])**.** *Every orthogonal TRS is confluent.* ∎

Similarly, criteria exist to detect termination of a TRS. Several are based on reduction orders.

Theorem 2.4.7. *A TRS* $\mathcal{R}$ *is terminating if and only if there exists a reduction order* $>$ *with* $\ell > r$ *for all* $\ell \to r \in \mathcal{R}$. ∎

We will introduce three orders to show termination. They are all instances of simplification orders.

Definition 2.4.8. *The strict* subterm relation $\rhd \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V}) \times \mathcal{T}(\mathcal{F}, \mathcal{V})$ *is defined as:* $s \rhd t$ *if there exists a position* $p \neq \varepsilon$ *such that* $s|_p = t$.

Definition 2.4.9. *A rewrite order* $>$ *is a* simplification order *if* $\rhd \subseteq >$.

Simplification orders ensure termination:

Theorem 2.4.10 ([20, 21])**.** *Let* $\mathcal{F}$ *be a finite signature and* $\mathcal{V}$ *a countable set of variables. Every simplification order on* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *is a reduction order.* ∎

In the following, a *precedence* on $\mathcal{F}$ is a strict ordering on $\mathcal{F}$.

Definition 2.4.11. *A* status τ *is a function that maps every* $f \in \mathcal{F}$ *to either* mul *or* lex_π. *Here* π *is a permutation on* $\{1,\dots,n\}$, *where* n *is the arity of* f.

Let $>$ *be a partial order on* $\mathcal{T}(\mathcal{F},\mathcal{V})$, *and* $f \in \mathcal{F}$ *be of arity* n. *The partial order* $>^{\tau(f)}$ *is defined on sequences of length* n *of terms as follows.*

$$(s_1,\dots,s_n) >^{\tau(f)} (t_1,\dots,t_n) \text{ is defined as } \begin{cases} \{\!\{s_1,\dots,s_n\}\!\} >^{mul} \{\!\{t_1,\dots,t_n\}\!\} & \text{if } \tau(f) = mul \\ (s_1,\dots,s_n) >^{lex} (t_{\pi(1)},\dots,t_{\pi(n)}) & \text{if } \tau(f) = lex_\pi. \end{cases}$$

Definition 2.4.12. *Let* $\mathcal{F}$ *be a signature,* $\mathcal{V}$ *a countable set of variables,* $\succ$ *be a well-founded precedence, and* τ *be a status. The* recursive path order $>_{rpo} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ *is defined as as* $s >_{rpo} t$ *if* s *has the form* $f(s_1,\dots,s_n)$ *and*

1. $s_i = t$ *or* $s_i >_{rpo} t$ *for some* $1 \le i \le n$, *or*
2. $t = g(t_1,\dots,t_m)$, $s >_{rpo} t_i$ *for all* $1 \le i \le m$, *and either*
   a) $f \succ g$ *or*
   b) $f = g$ *and* $(s_1,\dots,s_n) >_{rpo}^{\tau(f)} (t_1,\dots,t_m)$.

When requiring $\tau(f) = \text{mul}$ for all function symbols, the order is called the *multiset path order (MPO)*. It was originally introduced by Dershowitz [21]. The *lexicographic path order (LPO)*, introduced by Kamin and Lévy [45], is obtained by requiring $\tau(f) = \text{mul}_\pi$ for all function symbols. The recursive path order combines these two ideas of showing termination of TRSs.

Theorem 2.4.13 ([75])**.** *Let* $\mathcal{F}$ *be a signature,* $\mathcal{V}$ *be a countable set of variables,* $\succ$ *be a precedence, and* τ *be a status on* $\mathcal{F}$. *Then* $>_{rpo}$ *is a simplification order on* $\mathcal{T}(\mathcal{F},\mathcal{V})$. ∎

The next order is due to Knuth and Bendix [49].

Definition 2.4.14. *A* weight function *for a signature* $\mathcal{F}$ *is a tuple* (w, w_0), *where* w *is a function* $w : \mathcal{F} \to \mathbb{N}$ *and* $w_0 \in \mathbb{N}$ *is a constant such that* $w_0 > 0$ *and* $w(c) \ge w_0$ *for all constants* $c \in \mathcal{F}$. *Given a signature* $\mathcal{F}$ *and a weight-function* (w, w_0), *the* weight *of a term* $t \in \mathcal{T}(\mathcal{F},\mathcal{V})$ *is defined as:*

$$w(t) = \begin{cases} w_0 & \text{if } t \in \mathcal{V} \\ w(f) + \sum_{i=1}^{n} w(t_i) & \text{if } t = f(t_1,\dots,t_n) \end{cases}$$

A weight function (w, w_0) *is* admissible *for a precedence* $\succ$, *if* $\succsim g$ *for all function symbols* g *whenever* f *is a unary function symbol with* $w(f) = 0$.

In the next definition we write $\mathcal{F}^n$ for the set of all n-ary function symbols.

Definition 2.4.15. *Let $\succ$ be a precedence and (w, w_0) a weight function. The* Knuth-Bendix order (KBO) *is defined as $s >_{kbo} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either*

1. *$w(s) > w(t)$, or*
2. *$w(s) = w(t)$ and one of the following conditions hold:*
   a) *there is a function symbol $f \in \mathcal{F}^1$, a variable x and an integer n such that $s = f^n(x)$ and $t = x$, or*
   b) *$s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, and there exists an i with $1 \leq i \leq n$ such that $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$ and $s_i >_{kbo} t_i$, or*
   c) *$s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$ and $f \succ g$.*

Theorem 2.4.16 ([49])**.** *Let $\succ$ be a precedence on $\mathcal{F}$, and (w, w_0) and an admissible weight-function. Then $>_{kbo}$ generated by $\succ$ and (w, w_0) is a simplification order.* ∎

Sorted Term Rewriting Systems

Definition 2.4.17. *Let $\mathcal{S}$ be a non-empty set of* sorts*. An $\mathcal{S}$-sorted signature is a countable set of function symbols $\mathcal{F}$, where each $f \in \mathcal{F}$ of arity n is associated with the* function signature $\mathsf{sig}(f) \in \mathcal{S}^{n+1}$*, where* sig *is a function* $\mathsf{sig} : \mathcal{F} \to \mathcal{S}^*$*.*

Here components from 1 to n of sig provide the sort of each argument, and the $n+1$-th component is the sort of the result of the function. We will write $f : s_1 \times \cdots \times s_n \to s_{n+1}$ when f has the signature $(s_1, \ldots, s_{n+1})$.

Definition 2.4.18. *An $\mathcal{S}$-sorted set A is a family of sets $\{A_s\}_{s \in S}$. For an $\mathcal{S}$-sorted set $\mathcal{V}$ of variables with $\mathcal{V}_s \cap \mathcal{V}_t = \varnothing$ for all $s \neq t$, let $\mathcal{T}(\mathcal{F}, \mathcal{V})_s$ denote the set of terms with sort s over $\mathcal{F}$ and $\mathcal{V}$, defined inductively by*

$$\frac{x \in \mathcal{V}_s}{x \in \mathcal{T}(\mathcal{F}, \mathcal{V})_s} \qquad \frac{f \in \mathcal{F} \quad f : s_1 \times \cdots \times s_n \to s \quad t_i \in \mathcal{T}(\mathcal{F}, \mathcal{V})_{s_i}}{f(t_1, \ldots, t_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})_s}$$

This yields the $\mathcal{S}$-sorted set $\mathcal{T}(\mathcal{F}, \mathcal{V}) = \bigcup_{s \in \mathcal{S}} \mathcal{T}(\mathcal{F}, \mathcal{V})_s$. We associate with each term its sort*, i.e. the sort of a term t is s, if $t \in \mathcal{T}(\mathcal{F}, \mathcal{V})_s$. An $\mathcal{S}$-*sorted *TRS $\mathcal{R}$ is a TRS where for each rule $\ell \to r \in \mathcal{R}$, the terms ℓ and r are of the same sort.*

All notions for unsorted TRSs are lifted to sorted TRSs by limiting terms to well-sorted ones. Sorts are distinguished frequently in inductive theorem proving. In examples however we often consider TRSs with only one sort, and identify them with TRSs.

2.5 Equational Problems

Definition 2.5.1. *Let s and t be terms over $\mathcal{T}(\mathcal{F},\mathcal{V})$. An* equation *is a pair of terms (s,t). We usually write $s \approx t$. A set of equations is called an* equational system (ES).

Definition 2.5.2. *Let $\mathcal{F}$ be a signature. A tuple $\mathcal{A} = (A, \{f_\mathcal{A}\}_{f\in\mathcal{F}})$ is called an $\mathcal{F}$-*algebra. *Here*

- *$A \neq \varnothing$ is a set, called the* carrier *or* universe
- *$\{f_\mathcal{A}\}_{f\in\mathcal{F}}$ is a family of functions, called* interpretations of the function symbols. *Each f is assigned a function $f_\mathcal{A} : A^n \to A$, where n is the arity of f.*

Let $\alpha : \mathcal{V} \to A$ be a variable assignment. *An $\mathcal{F}$-algebra together with a variable assignment induces an* evaluation $[\alpha]_\mathcal{A}(t)$ *of a term t as follows*

$$[\alpha]_\mathcal{A}(t) = \begin{cases} \alpha(t) & \text{if } t \in \mathcal{V} \\ f_\mathcal{A}([\alpha]_\mathcal{A}(t_1), \ldots, [\alpha]_\mathcal{A}(t_n)) & \text{if } t = f(t_1,\ldots,t_n) \end{cases}$$

Definition 2.5.3. *Let $s \approx t$ be an equation over a signature $\mathcal{F}$.*

- *An $\mathcal{F}$-algebra $\mathcal{A} = (A, \{f_\mathcal{A}\}_{f\in\mathcal{F}})$* models *$s \approx t$, written $\mathcal{A} \models s \approx t$, if $[\alpha]_\mathcal{A}(s) = [\alpha]_\mathcal{A}(t)$ for all variable assignments α.*
- *An $\mathcal{F}$-algebra $\mathcal{A}$ is a* model of an ES *$\mathcal{E}$, written $\mathcal{A} \models \mathcal{E}$, if $\mathcal{A} \models s \approx t$ for all $s \approx t \in \mathcal{E}$.*
- *The equation $s \approx t$* follows *from an ES $\mathcal{E}$ (is* valid, *is* true *in an ES $\mathcal{E}$), written $\mathcal{E} \models s \approx t$, if $\mathcal{A} \models s \approx t$ for all algebras $\mathcal{A}$ with $\mathcal{A} \models \mathcal{E}$.*
- *The relation $\approx_\mathcal{E} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as $s \approx_\mathcal{E} t$ if $\mathcal{E} \models s \approx t$.*
- *Given an ES $\mathcal{E}$, and an equation $s \approx t$, the question whether $s \approx t$ follows from $\mathcal{E}$ is called* the word problem.

Definition 2.5.4. *The* equational step *relation $\to_\mathcal{E} \subseteq \mathcal{T}(\mathcal{F},\mathcal{V}) \times \mathcal{T}(\mathcal{F},\mathcal{V})$ is defined as $s \to_\mathcal{E} t$ if there exist a position $p \in \mathcal{P}\mathsf{os}(s)$, a substitution σ, and an equation $u \approx v \in \mathcal{E}$, such that $s|_p = u\sigma$ and $t = s[v\sigma]_p$. Here $\mathcal{E}$ is an ES. We say $s \approx t$ can be* deduced *from $\mathcal{E}$, if $s \leftrightarrow^*_\mathcal{E} t$.*

Theorem 2.5.5 (Birkhoff [14])**.** *Let $\mathcal{E}$ be an ES, and $s \approx t$ an equation. $\mathcal{E} \models s \approx t$ if and only if $s \leftrightarrow^*_\mathcal{E} t$.* ∎

Using Birkhoff's theorem, one can already construct a semi-decision procedure for $s \leftrightarrow^*_\mathcal{E} t$, since $\leftrightarrow^*_\mathcal{E}$ is recursively enumerable. In practice however, this approach does not work due to the sheer size of possible derivations. For example if two terms s and t are connected by ten equational steps, and if in each step four equations are applicable, the search tree has already $4^{10} = 1048576$ nodes. Here, equivalent complete TRSs come in handy.

Definition 2.5.6. *Let $\mathcal{E}$ be an ES and $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$ and set of variables $\mathcal{V}$. The TRS $\mathcal{R}$ is* equivalent *to $\mathcal{E}$ if $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\mathcal{E}}$.*

To test equivalence of $\leftrightarrow^*_{\mathcal{R}}$ and $\leftrightarrow^*_{\mathcal{E}}$, it suffices to check equivalence only w.r.t. to equations and rules:

Lemma 2.5.7. *Let $\mathcal{E}$ be an ES and $\mathcal{R}$ a TRS over a signature $\mathcal{F}$ and set of variables $\mathcal{V}$. Then $\mathcal{R}$ is equivalent to $\mathcal{E}$ if and only if $\ell \leftrightarrow^*_{\mathcal{E}} r$ holds for every $\ell \to r \in \mathcal{R}$, and $s \leftrightarrow^*_{\mathcal{R}} t$ holds for every $s \approx t \in \mathcal{E}$.* ∎

We will use the following notion throughout the rest of this thesis.

Definition 2.5.8. *A TRS $\mathcal{R}$ is* complete for an ES *$\mathcal{E}$, if $\mathcal{R}$ is complete and equivalent to $\mathcal{E}$.*

We now arrive at a central theorem, that builds the foundation for equational theorem proving.

Theorem 2.5.9. *Let $\mathcal{R}$ be a complete TRS for $\mathcal{E}$. Then $\mathcal{E} \models s \approx t$ iff $s{\downarrow_{\mathcal{R}}} = t{\downarrow_{\mathcal{R}}}$.*

Proof.

	$\mathcal{E} \models s \approx t$	
iff	$s \leftrightarrow^*_{\mathcal{E}} t$	by Theorem 2.5.5 (Birkhoff)
iff	$s \leftrightarrow^*_{\mathcal{R}} t$	by equivalence of $\mathcal{R}$ and $\mathcal{E}$
iff	$s{\downarrow_{\mathcal{R}}} = t{\downarrow_{\mathcal{R}}}$	by Theorem 2.1.9

∎

Definition 2.5.10. *Let $\mathcal{E}$ be an ES over a signature $\mathcal{F}$ and set of variables $\mathcal{V}$, and $s \approx t$ an equation. Then $s \approx t$ is an* inductive theorem *of $\mathcal{E}$, written $\mathcal{E} \vdash_i s \approx t$, if $\mathcal{E} \models s\sigma_g \approx t\sigma_g$ for all ground substitutions σ_g on $s \approx t$.*

Using Birkhoff's theorem, one can reduce the question $\mathcal{E} \vdash_{\mathrm{i}} s \approx t$ of inductive validity to the question whether $s\sigma_g \leftrightarrow^*_{\mathcal{E}} t\sigma_g$ holds for all substitutions σ_g that are ground both on s and t. However unlike for completion as in Theorem 2.5.9, there exists no definitive single characterization of a suitable TRS $\mathcal{R}$ for inductive theorem proving. The question on how to express the requirements of a suitable TRS $\mathcal{R}$ that ensures joinability on ground terms of a conjecture $s \approx t$ will be the subject of Chapter 5.

Chapter 3

Completion Procedures

Given an equational set $\mathcal{E}$, completion procedures try to construct a complete (confluent and terminating) TRS $\mathcal{R}$ that is equivalent for $\mathcal{E}$. By Theorem 2.4.5, confluence is decidable for terminating TRSs. Completion procedures thus try to first compute a terminating candidate TRS, and then test its confluence. Therefore ensuring termination is of central importance. All extensions of the original procedure have in common that they focus on and try to increase the capability of showing termination. In this chapter we briefly recall these extensions and illustrate their workings with small examples.

3.1 Knuth-Bendix Completion

Knuth and Bendix' completion procedure was the first to employ critical pairs to find a complete TRS. It was originally formulated in procedural form, which is listed as Algorithm 1.

The procedure takes as input not only an equational system $\mathcal{E}$ but also a reduction order $>$, and operates on a set of equations $\mathcal{E}_i$, initially $\mathcal{E}$, and a set of rules $\mathcal{R}_i$, initially empty. In each iteration, an equation from $\mathcal{E}_i$ is selected by a *selection heuristic*. Then both sides of the equation are rewritten to some normal form with respect to the current $\mathcal{R}_i$. If the normal forms are syntactically equal, the equation is removed from $\mathcal{E}_i$, and the iteration continues. Otherwise, if the

chosen normal forms for each side are incomparable, the algorithm stops with failure. If they are comparable in the ordering however, the original equation is removed from $\mathcal{E}_i$, the normalized equation is oriented, and critical pairs w.r.t. the new rule and $\mathcal{R}_i$ are added to $\mathcal{E}_i$. The new rule is added to $\mathcal{R}_i$ and the next iteration starts.

Algorithm 1 Knuth-Bendix Completion

Input: an equational system $\mathcal{E}$ and a reduction order $>$
Output: a TRS $\mathcal{R}$ or `FAIL`

1: **procedure** KB-COMPLETE
2: $\quad\mathcal{R}_0 := \varnothing$
3: $\quad\mathcal{E}_0 := \mathcal{E}$
4: $\quad i := 0$
5: $\quad$**while** $\mathcal{E}_i \neq \varnothing$ **do**
6: $\quad\quad$SELECT $s \approx t \in \mathcal{E}_i$
7: $\quad\quad s' \approx t' := s{\downarrow_{\mathcal{R}_i}} \approx t{\downarrow_{\mathcal{R}_i}}$
8: $\quad\quad$**if** $s' = t'$ **then**
9: $\quad\quad\quad\mathcal{R}_{i+1} := \mathcal{R}_i$
10: $\quad\quad\quad\mathcal{E}_{i+1} := \mathcal{E}_i \setminus \{s \approx t\}$
11: $\quad\quad$**else**
12: $\quad\quad\quad$**if** $s' > t'$ **then**
13: $\quad\quad\quad\quad\ell \to r := s' \to t'$
14: $\quad\quad\quad$**else**
15: $\quad\quad\quad\quad$**if** $t' > s'$ **then**
16: $\quad\quad\quad\quad\quad\ell \to r := t' \to s'$
17: $\quad\quad\quad\quad$**else**
18: $\quad\quad\quad\quad\quad$`FAIL`
19: $\quad\quad\quad\quad$**end if**
20: $\quad\quad\quad$**end if**
21: $\quad\quad\quad\mathcal{R}_{i+1} := \mathcal{R}_i \cup \{\ell \to r\}$
22: $\quad\quad\quad\mathcal{E}_{i+1} := \mathcal{E}_i \cup \mathsf{CP}(\mathcal{R}_i \cup \{\ell \to r\}, \{\ell \to r\})$
23: $\quad\quad$**end if**
24: $\quad\quad i := i + 1$
25: $\quad$**end while**
26: $\quad$**return** $\mathcal{R}$
27: **end procedure**

One should remark here, that when normalizing the selected equation (line 7), several possible normal forms can exist, since $\mathcal{R}_i$ is terminating, but needs not to be confluent. Also, according to the original description of the algorithm in [49], after constructing the new rule $\ell \to r$, all left and right hands sides of rules from $\mathcal{R}_i$ are rewritten to a normal form to remove redundant rules. This is called

inter-reduction. However, one needs extra care in this case, since applying inter-reduction before normalizing the selected equation (i.e. in-between line 6 and line 7) leads to an incorrect algorithm. Huet [40] was the first to give a full proof of soundness of the algorithm including inter-reduction.

Theorem 3.1.1. *Let $\mathcal{E}$ be an ES, and $>$ a reduction order. If* KB-Complete$(\mathcal{E}, >)$ *terminates with output $\mathcal{R}$, then $\mathcal{R}$ is a complete TRS for $\mathcal{E}$.* ∎

We give a brief example to illustrate the mechanisms of the algorithm, and also one of its weaknesses. To keep track of rules and equations, when referring to a rule originating from an indexed equation $j : s \approx t$, we write j to denote $s \to t$, and j' to denote $t \to s$. However we omit the counter i of the Knuth-Bendix algorithm for brevity.

Example 3.1.2. *Consider the ES $\mathcal{E}$ consisting of the equalities:*

$$1\colon\quad \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2\colon\quad \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3\colon\quad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

and the lexicographic path order $>$ with precedence $+ \succ \mathsf{s} \succ \mathsf{p}$. Suppose we select the first rule. Since there are no critical pairs of rule 1 on itself, we have $\mathcal{E} = \{2, 3\}$ and $\mathcal{R} = \{1\}$. We continue by selecting the second rule, which leads to $\mathcal{E} = \{3\}$ and $\mathcal{R} = \{1, 2\}$. Since the critical pairs $\mathsf{s}(x) \approx \mathsf{s}(x)$ and $\mathsf{p}(x) \approx \mathsf{p}(x)$ between 1 and 2 are trivial, we can ignore them, continue and select rule 3. There is one critical pair between rule 1 and rule 3

$$4\colon\quad x + y \approx \mathsf{s}(\mathsf{p}(x) + y)$$

which is in normal form w.r.t. $\mathcal{R}$. Thus $\mathcal{E} = \{4\}$ and $\mathcal{R} = \{1, 2, 3\}$. We select equation 4, and obtain three new critical pairs:

$$5\colon\quad \mathsf{p}(x + y) \approx \mathsf{p}(x) + y \qquad 6\colon\quad (x + y) + z \approx \mathsf{s}((\mathsf{p}(x) + y) + z)$$
$$7\colon\quad \mathsf{s}(x + y) \approx \mathsf{s}(x) + y$$

Suppose rule 5 is selected. Then $\mathcal{R} = \{1, 2, 3, 4', 5'\}$ is complete. After two iterations where equations 6 and 7 are selected, rewritten to identical normal forms and removed from $\mathcal{E}$, we arrive at $\mathcal{E} = \varnothing$ and the algorithm succeeds. It should be noted, that rule $4'$ is redundant, since both sides of it rewrite to identical normal form by $\mathcal{R} = \{1, 2, 3, 5'\}$.

It can be observed in Example 3.1.2 that the algorithm has some inefficiency, since it does not apply inter-reduction to remove redundant rules from the current TRS in every iteration. As mentioned, one has to be careful when rewriting rules of a TRS by itself, as it might affect soundness, cf. [75]. Bachmair et al. [12] made this condition explicit, by reformulating the procedure into a set of *inference rules* KB, which operate on a tuple $(\mathcal{E}, \mathcal{R})$ of an equational set and a TRS. To formulate the condition of soundness, the encompassment order is used.

DEDUCE	$\dfrac{\mathcal{E},\mathcal{R}}{\mathcal{E}\cup\{s\approx t\},\mathcal{R}}$	if $s\ {}_{\mathcal{R}}\!\leftarrow\cdot\rightarrow_{\mathcal{R}} t$
ORIENT	$\dfrac{\mathcal{E}\cup\{s\approx t\}}{\mathcal{E},\mathcal{R}\cup\{s\to t\}}$	if $s > t$
DELETE	$\dfrac{\mathcal{E}\cup\{s\approx s\},\mathcal{R}}{\mathcal{E},\mathcal{R}}$	
SIMPLIFY	$\dfrac{\mathcal{E}\cup\{s\approx t\},\mathcal{R}}{\mathcal{E}\cup\{u\approx t\},\mathcal{R}}$	if $s\to_{\mathcal{R}} u$
COMPOSE	$\dfrac{\mathcal{E},\mathcal{R}\cup\{s\to t\}}{\mathcal{E},\mathcal{R}\cup\{s\to u\}}$	if $t\to_{\mathcal{R}} u$
COLLAPSE	$\dfrac{\mathcal{E},\mathcal{R}\cup\{s\to t\}}{\mathcal{E}\cup\{u\approx t\},\mathcal{R}}$	if $s\xrightarrow{\sqsupset}_{\mathcal{R}} u$

Figure 3.1: Inference rules KB of Knuth-Bendix completion

Definition 3.1.3. *The* encompassment *quasi order $\sqsupseteq$ is defined as $s \sqsupseteq t$ if there exist a position $p \in \mathcal{P}\mathsf{os}(s)$ and a substitution σ, such that $s|_p = t\sigma$. Its strict part $\sqsupset$ is defined as $s \sqsupset t$ if $s \sqsupseteq t$ and $t \not\sqsupseteq s$. We will write $s \xrightarrow{\sqsupset}_{\mathcal{R}} t$ if $s \to_{\ell\to r} t$ for some $\ell \to r \in \mathcal{R}$ and $s \sqsupset \ell$.*

Note that the strict part of the encompassment order is well-founded.

Definition 3.1.4. *Let KB be the set of inference rules depicted in Figure 3.1. A* completion procedure *takes as input an ES $\mathcal{E}$ and a reduction order $>$, and generates a run with $\mathcal{E}_0 = \mathcal{E}$ and $\mathcal{R}_0 = \varnothing$. Here, a* run *is a sequence*

$$(\mathcal{E}_0,\mathcal{R}_0) \vdash_{KB} (\mathcal{E}_1,\mathcal{R}_1) \vdash_{KB} (\mathcal{E}_2,\mathcal{R}_2) \vdash_{KB} (\mathcal{E}_3,\mathcal{R}_3) \vdash_{KB} \dots$$

where for all $i \geq 0$ the first component $\mathcal{E}_i$ is an equational system, the second component $\mathcal{R}_i$ is a TRS and $(\mathcal{E}_{i+1},\mathcal{R}_{i+1})$ is reachable from $(\mathcal{E}_i,\mathcal{R}_i)$ by one of the inferences.

The output of the procedure is formulated as persistent sets.

Definition 3.1.5. *The sets of* persistent equations $\mathcal{E}_\omega$ *and* persistent rules $\mathcal{R}_\omega$ *are defined as*

$$\mathcal{E}_\omega = \bigcup_{i\geq 0}\bigcap_{j\geq i}\mathcal{E}_j \qquad \text{and} \qquad \mathcal{R}_\omega = \bigcup_{i\geq 0}\bigcap_{j\geq i}\mathcal{R}_j.$$

Definition 3.1.6. *The run generated by a completion procedure on* $\mathcal{E}$ fails *if* $\mathcal{E}_\omega \neq \varnothing$. *A run is* fair *if*

$$\mathsf{CP}(\mathcal{R}_\omega) \subseteq \bigcup_{i \geq 0} \mathcal{E}_i.$$

Theorem 3.1.7. *Suppose a completion procedure generates a non-failing and fair run on* $\mathcal{E}$. *Then* $\mathcal{R}_\omega$ *is complete for* $\mathcal{E}$. ∎

Thus Theorem 3.1.7 applies not just to one particular algorithm, but to every procedure that applies the inferences sequentially in a fair manner. Actual implementations of course need to turn the procedure into some algorithmic form, and usually implementations are similar to the original Knuth-Bendix completion or Huet's procedure [40].

3.2 Completion with Termination Tools

Knuth-Bendix completion requires to fix the ordering a priori. However the orders we presented in Chapter 2 are sometimes simply too weak to prove termination of a TRS. Consider for example the ES with just one equation

$$\mathsf{f}(x) * \mathsf{g}(y) \approx \mathsf{g}(y) * \mathsf{f}(x)$$

This equation appears for example in theories of commuting group endomorphisms [74]. Orienting the equation in either direction results in a complete TRS. But one can verify that there exists no LPO, MPO or KBO that allows to orient this rule. However much progress has been made in showing the termination of term rewriting systems automatically. Here we especially mention the dependency pair method [8, 30, 37] and matrix interpretations [23]. Both methods are able to show termination in the example.

Wehrman et al. [82] suggested to not fix a reduction order in advance, but instead to test termination of the TRS component $\mathcal{R}$ during completion and use $\to_{\mathcal{R}}^{+}$ as the ordering. In practice, the test for termination can be done by a dedicated termination tool. Note that $\to_{\mathcal{R}}^{+}$ is a reduction order whenever $\mathcal{R}$ is terminating. Based on this idea, they formulated a framework for completion based on inference rules, depicted in Figure 3.2.

The notion of run, fairness and failure can be extended in a straight-forward manner for the inference rules KBT. Initially the component C is empty.

Theorem 3.2.1 ([82])**.** *Suppose a completion procedure generates a non-failing and fair run with KBT on input* $\mathcal{E}$. *Then* $\mathcal{R}_\omega$ *is complete and equivalent for* $\mathcal{E}$. ∎

One might wonder why ensuring termination of $\mathcal{R}_i$ is not sufficient and why adding a dedicated component C is necessary. The role of C is to *incrementally*

DEDUCE	$\dfrac{\mathcal{E},\mathcal{R},C}{\mathcal{E}\cup\{s\approx t\},\mathcal{R},C}$	if $s\ {}_{\mathcal{R}}\!\leftarrow\cdot\rightarrow_{\mathcal{R}} t$
ORIENT	$\dfrac{\mathcal{E}\cup\{s\approx t\},C}{\mathcal{E},\mathcal{R}\cup\{s\rightarrow t\},C\cup\{s\rightarrow t\}}$	if $C\cup\{s\rightarrow t\}$ is terminating
DELETE	$\dfrac{\mathcal{E}\cup\{s\approx s\},\mathcal{R},C}{\mathcal{E},\mathcal{R},C}$	
SIMPLIFY	$\dfrac{\mathcal{E}\cup\{s\approx t\},\mathcal{R},C}{\mathcal{E}\cup\{u\approx t\},\mathcal{R},C}$	if $s\rightarrow_{\mathcal{R}} u$
COMPOSE	$\dfrac{\mathcal{E},\mathcal{R}\cup\{s\rightarrow t\},C}{\mathcal{E},\mathcal{R}\cup\{s\rightarrow u\},C}$	if $t\rightarrow_{\mathcal{R}} u$
COLLAPSE	$\dfrac{\mathcal{E},\mathcal{R}\cup\{s\rightarrow t\},C}{\mathcal{E}\cup\{u\approx t\},\mathcal{R},C}$	if $s\ \xrightarrow{\sqsupset}_{\mathcal{R}} u$

Figure 3.2: Inference rules KBT of Completion with Termination Tools

construct an order. Whenever a step $(\mathcal{E}_i,\mathcal{R}_i,C_i)\vdash_{\mathrm{KBT}}(\mathcal{E}_{i+1},\mathcal{R}_{i+1},C_{i+1})$ is applied, the inclusion $C_i\subseteq C_{i+1}$ holds by definition, and thus we have $\rightarrow^+_{C_i}\subseteq\rightarrow^+_{C_{i+1}}$. However due to the rule COLLAPSE, $R_i\subseteq R_{i+1}$ does not necessarily hold and thus $\rightarrow^+_{\mathcal{R}_i}\subseteq\rightarrow^+_{\mathcal{R}_{i+1}}$ also does not hold in general. The question whether changing the order in a non-incremental manner still yields a correct procedure was formulated as problem #35 in the RTA list of open problems[1], and finally answered negatively in [69].

To practically apply the inferences, Wehrman et al. proposed an algorithm very similar to Huet's [40].

3.3 Multi-Completion

As seen, in Knuth-Bendix completion, a fixed order is expected as an input parameter, and in completion with termination tools the order is constructed on-the-fly by orienting equations into a terminating TRS. If an equation can be oriented in both directions, a choice between the corresponding orders has to be made. Therefore in both procedures failure or success heavily depends on the choice of the reduction order.

[1] `http://rtaloop.mancoosi.univ-paris-diderot.fr`

Example 3.3.1. *Consider the following ES $\mathcal{E}$ from [72], consisting of*

$$1\colon \quad (x+y)+z \approx x+(y+z) \qquad\qquad 2\colon \quad \mathsf{f}(x+y) \approx \mathsf{f}(x)+\mathsf{f}(y)$$

Let $>_1$ be the lexicographic path order with precedence $\mathsf{f} \succ +$, and $>_2$ the lexicographic path order with precedence $+ \succ \mathsf{f}$. A run with $>_1$ succeeds with the complete and equivalent TRS $\{1,2\}$. However a run with $>_2$ fails, since then $\mathcal{E}_\omega$ is not empty.

In general it is not trivial to make a suitable choice for an order in advance. To practically apply completion with termination tools, Wehrman et al. [82] developed a heuristical *best-first* search strategy to select one orientation if both are possible. In the case of unorientable equations, backtracking is employed to recover failed attempts. One drawback of such an approach is that a complete TRSs is easily missed if infinitely many orientable equations are generated as in Example 3.3.1, or if the best-first heuristic dos not apply for the ES in question.

ORIENT
$$\frac{N \cup \{\langle s : t, L_1, L_2, L_3 \cup L\rangle\}}{N \cup \{\langle s : t, L_1 \cup L, L_2, L_3\rangle\}}$$
if $L \neq \varnothing, L_3 \cap L = \varnothing$ and $s >_i t$ for all $i \in L$

DELETE
$$\frac{N \cup \{\langle s : s, \varnothing, \varnothing, L\rangle\}}{N} \quad \text{if } L \neq \varnothing$$

REWRITE-1
$$\frac{N \cup \{\langle s : t, L_1, L_2, L_3\rangle\}}{N \cup \left\{\begin{array}{c}\langle s : t, L_1 \backslash L, L_2, L_3 \backslash L\rangle, \\ \langle s : u, L_1 \cap L, \varnothing, L_3 \cap L\rangle\end{array}\right\}}$$
if $\langle \ell : r, L, \cdot, \cdot\rangle \in N$, $(L_1 \cup L_3) \cap L \neq \varnothing$, $t \to_{\ell \to r} u$, and t and ℓ are variants

REWRITE-2
$$\frac{N \cup \{\langle s : t, L_1, L_2, L_3\rangle\}}{N \cup \left\{\begin{array}{c}\langle s : t, L_1 \backslash L, L_2 \backslash L, L_3 \backslash L\rangle, \\ \langle s : u, L_1 \cap L, \varnothing, (L_2 \cup L_3) \cap L\rangle\end{array}\right\}}$$
if $\langle \ell : r, L, \cdot, \cdot\rangle \in N$, $(L_1 \cup L_2 \cup L_3) \cap L \neq \varnothing$, and $t \to_{\ell \to r} u$, and $t \sqsupset \ell$

DEDUCE
$$\frac{N}{N \cup \{\langle s : t, \varnothing, \varnothing, L \cap L'\rangle\}}$$
if $\langle \ell : r, L, \cdot, \cdot\rangle \in N$, $\langle \ell' : r', L', \cdot, \cdot\rangle \in N$, $L \cap L' \neq \varnothing$, and $s \;{}_{\ell \to r}\!\leftarrow \cdot \to_{\ell' \to r'} t$

Figure 3.3: Inference rules MKB of Multi-Completion

A natural idea to handle Example 3.3.1 is to run Knuth-Bendix completion with several orders in parallel. Due to the sheer number of possible orders, this is usually not efficient. For example a signature with n function symbols induces $n!$ total precedences and thus at least $n!$ possible lexicographic path orderings, and even more if we count non-total precedences. Much performance can be gained, if operations involving equations and rules that are shared among runs with different orders have to be performed only once. Kurihara and Kondo [51] first introduced this idea, and central to its practical applicability is the introduction of a specialized data structure, a *node*.

Definition 3.3.2. *Let I be a set of indices, and $>_1, \ldots, >_n$ be n reduction orders. A* node *is a tuple $\langle s : t, L_1, L_2, L_3\rangle$, where $s : t$ is an ordered pair of terms, L_1, L_2 and L_3 are mutually disjoint subsets of I that satisfy the following two conditions. For $1 \leq i \leq n$:*

1. *$s >_i t$ whenever $i \in L_1$, and*
2. *$t >_i s$ whenever $i \in L_2$.*

A node $\langle s : t, L_1, L_2, L_3\rangle$ is identified with $\langle t : s, L_2, L_1, L_3\rangle$.

In practice, one could for example choose for the orders $>_1, \ldots >_n$ the lexicographic path orders generated by all possible precedences.

Definition 3.3.3. *Let $n = \langle s : t, L_1, L_2, L_3\rangle$ be a node and $i \in I$ an index. The* E-projection *and* R-projection *$E[n,i]$ and $R[n,i]$ of node n onto i are defined as*

$$E[n,i] = \begin{cases} \{s \approx t\} & \text{if } i \in L_3 \\ \varnothing & \text{otherwise} \end{cases} \qquad R[n,i] = \begin{cases} \{s \to t\} & \text{if } i \in L_1 \\ \{t \to s\} & \text{if } i \in L_2 \\ \varnothing & \text{otherwise} \end{cases}$$

They are extended to a set of nodes N by

$$E[N,i] = \bigcup_{n \in N} E[n,i] \qquad R[N,i] = \bigcup_{n \in N} R[n,i]$$

Inferences similar to Knuth-Bendix completion can be formulated. Here the inferences MKB, depicted in Figure 3.3, operate on nodes.

Definition 3.3.4. *A* run of multi-completion *is a sequence of sets of nodes $S = N_0 \vdash_{MKB} N_1 \vdash_{MKB} \ldots$ with $N_0 = \{\langle s : t, \varnothing, \varnothing, I\rangle \mid s \approx t \in \mathcal{E}\}$ such that N_{i+1} is reachable from N_i by on of the inferences of MKB.*

One can project a run of multi-completion to a run of Knuth-Bendix completion as follows. Given a run S of multi-completion and index $i \in I$ of the run, the *projection* $\mathrm{KB}[S,i]$ is generated as follows: First, every set of nodes N_j is projected to the tuple $(E[N_j,i], R[N_j,i])$. Then we yield a sequence of configurations,

where each $(E[N_{j+1}, i], R[N_{j+1}, i])$ is reachable from $(E[N_j, i], R[N_j, i])$ either by a step of Knuth-Bendix completion or by an equivalent step, i.e. both configurations are identical. Now all such equivalent steps with $(E[N_j, i], R[N_j, i]) = (E[N_{j+1}, i], R[N_{j+1}, i])$ are removed to obtain the projection $\mathrm{KB}[S, i]$.

Lemma 3.3.5. *Let S be a run of multi-completion, and $i \in I$. Then $KB[S, i]$ is a run of completion with KB.* ∎

Definition 3.3.6. *An MKB-completion procedure is a program that takes as input an ES $\mathcal{E}$ and an indexed set of orders I, and generates a run of multi-completion S. The set of* persisting nodes *N_ω is defined as*

$$N_\omega = \bigcup_{j \geq 0} \bigcap_{k \geq j} N_k.$$

A run of MKB fails, if $KB[S, i]$ fails for all $i \in I$. An MKB-completion is fair, *if it is finite and $KB[S, i]$ is non-failing for some some $i \in I$ or if it is infinite and $KB[S, i]$ is either fair or failing for all for all $i \in I$.*

The above definition follows [85] and differs from [51] to ensure soundness and completeness properties similar to Knuth-Bendix completion. We refer to [85] for a more in-depth discussion.

Theorem 3.3.7. *Let $\mathcal{E}$ be an ES, and I an indexed set of reduction orders. Let $S = N_0 \vdash_{MKB} N_1 \vdash_{MKB} \ldots$ be a non-failing, fair run of MKB-completion. Then there exists an $i \in I$ such that $R[N_\omega, i]$ is complete and equivalent for $\mathcal{E}$.* ∎

3.4 Multi-Completion with Termination Tools

Implementations of Multi-completion do not suffer from the indeterminism of the inference Orient of Knuth-Bendix completion and completion with termination tools. On the other hand, the latter significantly enhances the power of completion by making a larger class of reduction orders available. It was shown in [68] how these two approaches can be combined with the inference rules MKBTT, depicted in Figure 3.4.

While multi-completion considers indices of runs with different reduction orders, here runs of parallel completion procedures are directly identified by bitstrings. Whenever an equation can be oriented both ways, the run is split by creating two copies of all nodes and attaching a 0 resp. 1 to each of them. Intuitively a node $\langle s : t, R_1, R_2, E, C_1, C_2 \rangle$ is a tuple containing the equation itself, the set R_1 which collects all bitstrings where the equation is oriented $s \to t$, and the set R_2 which collects those where $t \to s$. Similar to KBT, the sets C_1 and C_2 collect constraints for R_1 and R_2 to ensure the implied orders are constructed in

Orient
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{\text{split}(N) \cup \{\langle s : t, R_1 \cup R_{lr}, R_2 \cup R_{rl}, E', C_1 \cup R_{lr}, C_2 \cup R_{rl}\rangle\}}$$
with $E_{lr}, E_{rl} \subseteq E$ such that

- $E_{lr} \cup E_{rl} \neq \varnothing$,
- $P = E_{lr} \cap E_{rl}$,
- $E' = E \backslash (E_{lr} \cup E_{rl})$
- $C[N, p] \cup \{s \to t\}$ terminates for all $p \in E_{lr}$,
- $C[N, p] \cup \{t \to s\}$ terminates for all $p \in E_{rl}$,
- $R_{lr} = (E_{lr} \backslash E_{rl} \cup \{p0 \mid p \in P\}$, and
- $R_{rl} = (E_r \backslash E_{lr}) \cup \{p1 \mid p \in P\}$,

and $\text{split}_p(N)$ replaces every $p \in P$ in any label in N by $p0$ and $p1$.

Delete
$$\frac{N \cup \{\langle s : s, \varnothing, \varnothing, E, \varnothing, \varnothing\rangle\}}{N}$$

Gc
$$\frac{N \cup \{\langle s : t, \varnothing, \varnothing, \varnothing, \varnothing\rangle\}}{N}$$

Rewrite-1
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{N \cup \left\{\begin{array}{c}\langle s : t, R_1 \backslash R, R_2, E \backslash R, C_1, C_2\rangle, \\ \langle s : t, R_1 \cap R, \varnothing, E \cap R, C_1, C_2\rangle\end{array}\right\}}$$
if $\langle \ell : r, R, \cdots\rangle \in N$, $t \to_{\ell \to r} u$, t and ℓ are variants, $R \cap (R_1 \cup E) \neq \varnothing$

Rewrite-2
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle\}}{N \cup \left\{\begin{array}{c}\langle s : t, R_1 \backslash R, R_2 \backslash R, E \backslash R, C_1, C_2\rangle, \\ \langle s : t, R_1 \cap R, \varnothing, (R_2 \cup E) \cap R, \varnothing, \varnothing\rangle\end{array}\right\}}$$
if $\langle \ell : r, R, \cdots\rangle \in N$, $t \to_{\ell \to r} u$, $t \sqsupset \ell$ and $R \cap (R_1 \cup R_2 \cup E) \neq \varnothing$

Deduce
$$\frac{N}{N \cup \{\langle s : t, \varnothing, \varnothing, R \cap R', \varnothing, \varnothing\rangle\}}$$
if there exist nodes $\langle \ell : r, R, \cdots\rangle, \langle \ell' : r', R', \cdots\rangle \in N$ such that $s \;{}_{\ell \to r}\!\leftarrow \cdot \to_{\ell' \to r'} t$ and $R \cap R' \neq \varnothing$

Subsume
$$\frac{N \cup \{\langle s : t, R_1, R_2, E, C_1, C_2\rangle, \langle s' : t', R_1', R_2', E', C_1', C_2'\rangle\}}{N \cup \{\langle s : t, R_1 \cup R_1', R_2 \cup R_2', E'', C_1 \cup C_1', C_2 \cup C_2'\rangle\}}$$
if $E'' = (E \backslash (R_1' \cup R_2' \cup C_1' \cup C_2')) \cup (E' \backslash (R_1 \cup R_2 \cup C_1 \cup C_2))$ and s and s', and t and t' are variants

Figure 3.4: Inference rules MKBTT of Multi-Completion w/ Termination-Tools

an incremental manner. The set E contains the indices of those runs, where the equation $s \approx t$ has been deduced, but not been oriented or rewritten to identity yet.

Soundness can be established in a similar way as done for multi-completion. For precise definitions and proofs, we refer to [67], and instead we illustrate how the inferences of MKBTT operate on the ES of Example 3.1.2.

Example 3.4.1 (continued from Example 3.1.2)**.** *Initially, three nodes are created:*

$$\begin{aligned} 1: &\quad \langle \mathsf{s}(\mathsf{p}(x)) : x, \varnothing, \varnothing, \{\varepsilon\}, \varnothing, \varnothing \rangle \\ 2: &\quad \langle \mathsf{p}(\mathsf{s}(x)) : x, \varnothing, \varnothing, \{\varepsilon\}, \varnothing, \varnothing \rangle \\ 3: &\quad \langle \mathsf{s}(x) + y : \mathsf{s}(x + y), \varnothing, \varnothing, \{\varepsilon\}, \varnothing, \varnothing \rangle \end{aligned}$$

After three applications of Orient*, we yield the following set of nodes. Note that orienting node* 3 *causes a split.*

$$\begin{aligned} 1: &\quad \langle \mathsf{s}(\mathsf{p}(x)) : x, \{0,1\}, \varnothing, \varnothing, \{0,1\}, \varnothing \rangle \\ 2: &\quad \langle \mathsf{p}(\mathsf{s}(x)) : x, \{0,1\}, \varnothing, \varnothing, \{0,1\}, \varnothing \rangle \\ 3: &\quad \langle \mathsf{s}(x) + y : \mathsf{s}(x + y), \{0\}, \{1\}, \varnothing, \{0\}, \{1\} \rangle \end{aligned}$$

Applying Deduce *to nodes* 1 *and* 3 *and to nodes* 2 *and* 3 *yields two new nodes:*

$$\begin{aligned} 4: &\quad \langle x + y : \mathsf{s}(\mathsf{p}(x) + y), \varnothing, \varnothing, \{0\}, \varnothing, \varnothing \rangle \\ 5: &\quad \langle \mathsf{p}(\mathsf{s}(x) + y) : x + y, \varnothing, \varnothing, \{1\}, \varnothing, \varnothing \rangle \end{aligned}$$

Applying Orient *on* 4 *and* 5*, modifies these nodes to*

$$\begin{aligned} 4: &\quad \langle x + y : \mathsf{s}(\mathsf{p}(x) + y), \varnothing, \{0\}, \varnothing, \varnothing, \{0\} \rangle \\ 5: &\quad \langle \mathsf{p}(\mathsf{s}(x) + y) : x + y, \{1\}, \varnothing, \varnothing, \{1\}, \varnothing \rangle \end{aligned}$$

The inference Deduce *on nodes* 4 *and* 2*, and on node* 5 *and* 1 *yield one new node:*

$$6: \quad \langle \mathsf{p}(x) + y : \mathsf{p}(x + y), \varnothing, \varnothing, \{0,1\}, \varnothing, \varnothing \rangle$$

Then Orient *modifies this equation to*

$$6: \quad \langle \mathsf{p}(x) + y : \mathsf{p}(x + y), \varnothing, \{0\}, \{1\}, \varnothing, \varnothing \rangle$$

Now Rewrite-2 *can be applied twice on node* 4*, first using node* 6 *and then node* 1*, and finally* Delete *removes node* 4 *entirely. We can now consider the run indexed by* 0*. The E-projection of index* 0 *consists of the equations of those nodes, where* 0 *appears in the E-component (fourth component) of a node — it is empty.*

The R-projection of a node n with respect to index 0 *is defined as* $\{s \to t\}$ *if* $p \in R_1$*, as* $\{t \to s\}$ *if* $p \in R_2$ *and* $\varnothing$ *otherwise. Thus the union of R-projections of all nodes yields the TRS* $\mathcal{R} = \{1, 2, 3, 6\}$*. Moreover one can verify that the run indexed by* 0 *is fair, since since all non-trivial critical pairs of* $\mathcal{R}$ *have been considered. Thus* $\mathcal{R}$ *is complete.*

Note how in completion with MKBTT of Example 3.4.1, several equations (equations 1, 2, 3 and 6) are shared among the runs. When the completion of an equational system is more complex than in the considered example, this effects a significant performance gain. Moreover MKBTT allows to fully use the progress made in research about automated termination analysis, and yields a very powerful procedure.

In the example, the sequence of selection of nodes and inferences on them leads quickly to a complete TRS. Two issues remain in this framework. The first one is that when practically applying this framework, the indeterminism of node-selection has to be resolved. Similar to the choice of the reduction order, the question which node to select when, can have an influence both on the outcome of the algorithm itself and on its performance. Secondly, the incorporation of implementational details, the data structure node, makes the formulation of the inferences more complex, and the extraction of the essential soundness properties of completion non-trivial.

3.5 Inductive Proofs by Completion

Methods to prove inductive theorem can roughly be classified into three categories. *Explicit methods* [15] construct an explicit induction scheme that is adjusted specifically for the equational system and the conjecture in question. *Proofs by consistency* [18] do not rely on an induction scheme at all. Instead they start a proof attempt directly and try to find a suitable induction scheme during the proof. Lastly, *implicit induction techniques* are somewhat between the two above. They are direct adaptations of completion to the setting of inductive theorem proving. While no explicit induction scheme is generated a priori, an induction scheme is implicitly provided by a *syntactic* order. Since the order changes the induction scheme, the choice of it affects the outcome of a proof attempt, quite similar to completion.

Due to the deep connection with completion we solely focus on the latter. For an overview of methods based on proofs by consistency, and a more in-depth discussion about the differences to implicit methods we refer to [18]. We remark that [18] refers to methods using proofs by consistency as *inductionless induction*. The terminology inductionless induction has been used by multiple authors in several different manners.

We will use the terminology inductionless induction in a different fashion than [18]. With it we refer to one specific implicit proof method due to Gramlich [34]. It greatly illustrates the connection between Knuth-Bendix completion and inductive proof methods. The second result we recall here is *rewriting induction*. Although there is no direct historical connection, it can be seen as a refinement of Gramlich's method. Several extensions to rewriting induction have been

$$\textsc{Delete}\quad \frac{\mathcal{E} \cup \{s \approx s\}, \mathcal{H}}{\mathcal{E}, \mathcal{H}}$$

$$\textsc{Simplify}\quad \frac{\mathcal{E} \cup \{s \approx t\}, \mathcal{H}}{\mathcal{E} \cup \{u \approx t\}, \mathcal{H}} \qquad \text{if } s \to_{\mathcal{R} \cup \mathcal{H}} u$$

$$\textsc{Expand}\quad \frac{\mathcal{E} \cup \{s \approx t\}, \mathcal{H}}{\mathcal{E} \cup \mathrm{Expd}(\mathcal{R}, \{s \to t\}), \mathcal{H} \cup \{s \to t\}} \qquad \text{if } \{s \to t\} \text{ is } \mathcal{R}\text{-expandable}$$

Figure 3.5: Inference rules of Rewriting Induction

proposed [1, 2], which allow to construct powerful inductive theorem provers. Multi-completion also has been adapted to be used with rewriting induction [66]. Such extensions however are beyond the scope of this thesis.

Before continuing, we first recall some definitions related to inductive theorems.

Definition 3.5.1. *A term t is $\mathcal{R}_0$-inductively-reducible, if for all ground substitutions σ_g on t, the term $t\sigma_g$ is $\mathcal{R}_0$-reducible. A TRS $\mathcal{R}$ is left-$\mathcal{R}_0$-inductively-reducible, if for all rules $\ell \to r \in \mathcal{R}$, the term ℓ is $\mathcal{R}_0$-inductively-reducible.*

Below when writing $\mathcal{R}_0 \vdash_{\mathrm{i}} \mathcal{H}$ for TRS $\mathcal{R}$ and $\mathcal{H}$ the latter is regarded as an ES. The next theorem is the basis of *inductionless induction*.

Theorem 3.5.2 ([34])**.** *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{H}$ be a terminating, and left-$\mathcal{R}_0$-inductively-reducible TRS. If $\mathsf{CP}(\mathcal{R}_0, \mathcal{H}) \subseteq {\downarrow_{\mathcal{R}}}$, then $\mathcal{R}_0 \vdash_i \mathcal{H}$.* ∎

It is not difficult to design inference rules for inductive proofs based on Theorem 3.5.2 that delete identities, rewrite equations or add critical pairs, see [34]. We skip such a calculus however and proceed to *rewriting induction* [62]. It relaxes the joinability conditions of inductionless induction. Instead of requiring all critical pairs to be joinable, joinability of only a specific subset suffices.

We recall some additional definitions. Given a TRS $\mathcal{R}$, the set $\mathcal{D} = \{\mathsf{root}(\ell) \mid \ell \to r \in \mathcal{R}\}$ is the set of *defined function symbols* (of $\mathcal{R}$). The set $\mathcal{C} = \mathcal{F} \setminus \mathcal{D}$ is the set of *constructor symbols*. A term $t = f(c_1, \ldots, c_n)$ is *basic*, if $f \in \mathcal{D}$ and $c_i \in \mathcal{T}(\mathcal{C}, \mathcal{V})$ for all $1 \leq i \leq n$. We write $\mathcal{B}(t)$ for the set of all *basic positions* $\{p \in \mathcal{P}\mathsf{os}(t) \mid t|_p \text{ is basic}\}$. A TRS is *quasi-reducible* if no ground basic term is in normal form. Let $\mathcal{R}_0$ be a quasi-reducible TRS with defined function symbols $\mathcal{D}$ and constructor symbols $\mathcal{C}$, and let $\mathcal{E}$ be an ES. We say that $\mathcal{R}$ is $\mathcal{R}_0$-*expandable* if for every $\ell \to r \in \mathcal{R}$ there exists a basic (w.r.t. $\mathcal{D}$ and $\mathcal{C}$) position in ℓ. We write $\mathrm{Expd}(\mathcal{R}_0, \mathcal{R})$ for the set of all critical pairs originating from overlaps $(\ell_1 \to$

$r_1, p, \ell_2 \to r_2)_\mu$ of $\mathcal{R}_0$ on $\mathcal{R}$, where p is a fixed (depending on $\ell_1 \to r_1$) basic position in ℓ_2.

Before continuing, we remark the following. Quasi-reducibility ensures that all instantiations of variables can be considered to be solely built upon constructor symbols. However quasi-reducibility rarely holds for unsorted TRSs. Thus one often assumes TRSs to be sorted when reasoning with inductive theorems. We will follow this approach throughout this thesis.

Theorem 3.5.3 ([62]). *Let $\mathcal{R}$ and $\mathcal{H}$ be TRSs such that $\mathcal{R}$ is quasi-reducible and $\mathcal{R} \cup \mathcal{H}$ is terminating. If $\mathrm{Expd}(\mathcal{R}, \mathcal{H}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{H}}$ then $\mathcal{R} \vdash_i \mathcal{H}$.* ∎

Inference rules for Rewriting Induction are depicted in Figure 3.5. We write $(\mathcal{E}, \mathcal{H}) \rightsquigarrow (\mathcal{E}', \mathcal{H}')$ when applying an inference rule of Figure 3.5. The reflexive transitive closure of application of such rules is depicted by $\rightsquigarrow^*$. Using Theorem 3.5.3 one can show correctness of deriving proofs by this inference system.

Theorem 3.5.4 ([62]). *Let $\mathcal{R}$ be a quasi-reducible TRS, $\mathcal{E}$ a set of equations and $>$ a reduction order with $\mathcal{R} \subseteq >$. If there exists a sequence of inferences and a TRS $\mathcal{H}$ such that $(\mathcal{E}, \varnothing) \rightsquigarrow^* (\varnothing, \mathcal{H})$, then $\mathcal{R} \vdash_i \mathcal{H}$.* ∎

To conclude this section, we give an example to show how a proof of a valid conjecture is obtained using the inference rules in Figure 3.5.

Example 3.5.5. *Consider the TRS $\mathcal{R}$ consisting of*

$$1\colon \quad 0 + y \to y \qquad\qquad 2\colon \quad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

with sorts $0 : \mathrm{nat}, \mathsf{s} : \mathrm{nat} \to \mathrm{nat}$ and $+ : \mathrm{nat} \times \mathrm{nat} \to \mathrm{nat}$. Let the set of conjectures $\mathcal{E}$ consist of the next equation.

$$3\colon \quad (x + y) + z \approx (x + y) + z$$

The proof starts by $(\{3\}, \varnothing)$. After expansion at the leftmost, innermost basic position we reach $(\{4, 5\}, \{3\})$ where 4 and 5 are the equations:

$$4\colon \quad y_1 + z \approx 0 + (y_1 + z) \qquad 5\colon \quad \mathsf{s}(x_2 + y_2) + z \approx \mathsf{s}(x_2) + (y_2 + z)$$

Both sides of equation 4 and 5 rewrite to identical normal form by the TRS $\{1, 2, 3\}$. Therefore by several steps of the rules Simplify *and* Delete *we have $(\{4, 5\}, \{3\}) \rightsquigarrow^* (\varnothing, \{3\})$. Thus $\mathcal{R} \vdash_i \mathcal{E}$.*

Chapter 4

Maximal Completion

In Chapter 3 we presented the original Knuth-Bendix completion procedure and several extensions. Searching for a complete TRS, these procedures face a kind of dilemma: With simplicity of the framework comes a decrease in power, and with more power comes an increase in the complexity of the framework due to incorporation of algorithmic and implementational details. The underlying reason is that all extensions essentially rely on the original algorithmic design of Knuth-Bendix' procedure to ensure soundness. This design imposes a certain way how to construct the search space, and a dedicated search algorithm to resolve all indeterminism.

During the early 2000's, much progress has been made in the area of constraint solving. Problems such as SAT, which are NP-hard and were thought to be intractable, turned out to be effectively solvable in many instances. Highly optimized algorithms, usually extensions of the DPLL-procedure [19], enabled applications in other domains, where encoding the problem to constraints and using a constraint solver has surpassed the development of dedicated algorithms. Such an approach is usually more efficient, and moreover allows to abstract implementational details from a framework as constraints. For example many techniques to automatically show termination of a TRS can be encoded as an instance of SAT.

To encode a problem as an instance of SAT, the search space has to be made

explicit in order to be representable by a finite number of constraints. Unfortunately, due to the complexity of the completion problem, it is non-trivial to explicitly give the search space that includes all desirable complete TRSs. But we present a new framework for completion by giving an encoding to a maximality problem over constraints. This allows to separate the soundness properties of completion from the search problem. Since the search space is represented indirectly by maximality constraints, no dedicated algorithm is needed any more, and implementational details are no longer part of the framework. Therefore it can be formulated in a much simpler way than existing powerful completion procedures, even though all existing completion procedures can be expressed in our framework by choosing suitable parameters. Experimental evaluation of a prototype implementation shows that the procedure can cope with and often surpasses existing state-of-the-art tools. Moreover due to its modularity, the framework can benefit from any future progress in MaxSAT or MaxSMT.

To formulate our framework we first look again at the completion procedure itself, and reformulate it as a maximality problem in Section 4.1. We show how to automate this formulation by giving an encoding to maximality constraints in Section 4.2, and we relate our framework to existing procedures in Section 4.3. Finally we evaluate our approach experimentally in Section 4.4.

4.1 Maximality

The task of completion is to find a complete TRS with respect to some *given* system of equalities. This notion of completeness is captured in the next lemma.

Lemma 4.1.1. *A TRS $\mathcal{R}$ is complete for an ES $\mathcal{E}$ if and only if $\mathcal{R}$ is terminating, $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^*$, and $\mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$.*

Proof. For the "if"-direction, by Knuth and Bendix' confluence criterion [49] (Theorem 2.4.5), confluence of $\mathcal{R}$ follows from $\mathsf{CP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$ and termination of $\mathcal{R}$. Moreover, $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$ and $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^*$ yield $\leftrightarrow_{\mathcal{R}}^* = \leftrightarrow_{\mathcal{E}}^*$. The "only if"-direction is immediate from $\leftrightarrow_{\mathcal{E}}^* = \leftrightarrow_{\mathcal{R}}^* \subseteq \downarrow_{\mathcal{R}}$. ∎

Lemma 4.1.1 yields a simple completion procedure. Let $\mathcal{E}$ be an ES. We assume that two parameter functions $\mathfrak{R}$ and S are given and the next two conditions hold for every ES $\mathcal{C}$: $S(\mathcal{C})$ is a set of equalities in $\leftrightarrow_{\mathcal{E}}^*$, and $\mathfrak{R}(\mathcal{C})$ is a set of terminating TRSs $\mathcal{R}$ with $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^*$.

Definition 4.1.2. *Given ESs $\mathcal{E}$ and $\mathcal{C}$, the procedure φ is defined as*

$$\varphi(\mathcal{C}) = \begin{cases} \mathcal{R} & \text{if } \mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}} \text{ for some } \mathcal{R} \in \mathfrak{R}(\mathcal{C}) \\ \varphi(\mathcal{C} \cup S(\mathcal{C})) & \text{otherwise} \end{cases}$$

Note that $\varphi(\mathcal{C})$ is neither unique nor defined in general.

Theorem 4.1.3. *$\varphi(\mathcal{E})$ is a complete TRS for an ES $\mathcal{E}$, if it is defined.* ∎

The procedure φ repeatedly expands $\mathcal{C}$ (initially $\mathcal{E}$) by $S(\mathcal{C})$ until $\mathfrak{R}(\mathcal{C})$ contains a complete TRS for $\mathcal{E}$. For its success the choice of $\mathfrak{R}(\mathcal{C})$ and $S(\mathcal{C})$ is crucial. Let $t{\downarrow_{\mathcal{R}}}$ denote a fixed normal form of t with respect to $\mathcal{R}$. We say that a TRS $\mathcal{R}$ is *over* an ES $\mathcal{C}$ if $\mathcal{R} \subseteq \mathcal{C} \cup \mathcal{C}^{-1}$. The set of all terminating TRSs over $\mathcal{C}$ is denoted by $\mathfrak{T}(\mathcal{C})$. We propose to use

$$\begin{aligned}\mathfrak{R}(\mathcal{C}) &= \mathit{Max}\,\mathfrak{T}(\mathcal{C})\\ S(\mathcal{C}) &= \bigcup_{\mathcal{R}\in\mathfrak{R}(\mathcal{C})} \{s{\downarrow_{\mathcal{R}}} \approx t{\downarrow_{\mathcal{R}}} \mid s \approx t \in \mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \text{ and } s{\downarrow_{\mathcal{R}}} \neq t{\downarrow_{\mathcal{R}}}\}\end{aligned}$$

Here *Max* computes all maximal sets of rewrite rules (called *maximal* TRSs) in its given family of TRSs, and this is the reason that we call our method *maximal* completion. Part (b) in the next lemma explains why *non*-maximal TRSs in $\mathfrak{T}(\mathcal{R})$ can be ignored safely.

Lemma 4.1.4. *Let $\mathcal{E}$ be an ES. The following claims hold:*

(a) Let $\mathcal{R}$ and $\mathcal{R}'$ be TRSs such that $\mathcal{R} \subseteq \mathcal{R}' \subseteq \leftrightarrow_{\mathcal{E}}^*$ with $\mathcal{R}'$ terminating. Then $\mathcal{R}'$ is complete for $\mathcal{E}$ if $\mathcal{R}$ is complete for $\mathcal{E}$.

(b) $\mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}}}$ for some $\mathcal{R} \in \mathfrak{T}(\mathcal{C})$ iff $\mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \subseteq {\downarrow_{\mathcal{R}}}$ for some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$.

Proof. (a) Due to completeness of $\mathcal{R}$, we have $\mathcal{E} \cup \mathsf{CP}(\mathcal{R}') \subseteq {\downarrow_{\mathcal{R}}}$. The claim follows together with $\to_{\mathcal{R}} \subseteq \to_{\mathcal{R}'}$. (b) The 'only if'-direction is straightforward from the first claim, and the converse is trivial. ∎

We illustrate maximal completion with an example.

Example 4.1.5. *Consider the ES $\mathcal{E}$ consisting of the equalities:*

$$1\colon\ \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2\colon\ \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3\colon\ \mathsf{s}(x)+y \approx \mathsf{s}(x+y)$$

We compute $\varphi(\mathcal{E})$ with the above $S(\mathcal{C})$ and $\mathfrak{R}(\mathcal{C})$.

(i) $\mathfrak{R}(\mathcal{E})$ consists of two TRSs $\{1,2,3\}$ and $\{1,2,3'\}$. Since the join-condition of φ does not hold, we have $\varphi(\mathcal{E}) = \varphi(\mathcal{E} \cup S(\mathcal{E}))$. Here $S(\mathcal{E})$ consists of two equalities:

$$4\colon\ x+y \approx \mathsf{s}(\mathsf{p}(x)+y) \qquad 5\colon\ \mathsf{p}(\mathsf{s}(x)+y) \approx x+y$$

(ii) $\mathfrak{R}(\{1,\dots,5\})$ consists of the two TRSs $\{1,2,3,4',5\}$ and $\{1,2,3',4',5\}$. Again the join-condition does not hold. Thus, $\varphi(\{1,\dots,5\}) = \varphi(\{1,\dots,9\})$, where

$$\begin{aligned} &6\colon\ (x+y)+z \approx \mathsf{s}((\mathsf{p}(x)+y)+z) && 7\colon\ \mathsf{p}((\mathsf{s}(x)+y)+z) \approx (x+y)+z\\ &8\colon\ \mathsf{p}(x)+y \approx \mathsf{p}(x+y) && 9\colon\ \mathsf{p}((x+y)+z) \approx (\mathsf{p}(x)+y)+z\end{aligned}$$

(iii) $\mathfrak{R}(\{1,\dots,9\})$ *consists of four TRSs including the TRS* $\mathcal{R}$ *of*

$$\{1,2,3,4',5,6',7,8,9'\}$$

which satisfies the join-condition. Thus, $\varphi(\{1,\dots,9\}) = \mathcal{R}$.

Hence, $\varphi(\mathcal{E}) = \mathcal{R}$ *and it is a complete TRS for* $\mathcal{E}$.

Very often a complete TRS resulting from maximal completion contains many superfluous rules. It is known that this problem is resolved by computing reduced TRSs (cf. [40]). A TRS $\mathcal{R}$ is *reduced* if $\ell \in \mathsf{NF}(\mathcal{R} \setminus \{\ell \to r\})$ and $r \in \mathsf{NF}(\mathcal{R})$ for all rules $\ell \to r \in \mathcal{R}$. We write $\widehat{\mathcal{R}}$ for the reduced TRS

$$\{\ell \to r \in \widetilde{\mathcal{R}} \mid \ell \in \mathsf{NF}(\widetilde{\mathcal{R}} \setminus \{\ell \to r\})\}$$

where $\widetilde{\mathcal{R}} = \{\ell \to r{\downarrow_{\mathcal{R}}} \mid \ell \to r \in \mathcal{R}\}$. The TRS $\widehat{\mathcal{R}}$ fulfils the desired property:

Lemma 4.1.6. *If a TRS* $\mathcal{R}$ *is complete for* $\mathcal{E}$*, then* $\widehat{\mathcal{R}}$ *is complete for* $\mathcal{E}$.

Proof. Using the fact that $\widehat{\mathcal{R}}$ is complete and $\leftrightarrow^*_{\mathcal{R}} = \leftrightarrow^*_{\widehat{\mathcal{R}}}$ (see [54]). ∎

Example 4.1.7 (continued from Example 4.1.5)**.** *We have* $\widetilde{\mathcal{R}} = \{1,\dots 9\}$. *The left-hand sides of rules* 5 *and* 7 *are reducible by rule* 3*, and the left-hand sides of rules* $4', 6'$ *and* $9'$ *are reducible by rule* 8*. Rules* $i = 1,2,3,8$ *are in normal form with respect to* $\widetilde{\mathcal{R}} \setminus \{i\}$. *Thus the reduced version* $\widehat{\mathcal{R}}$ *is* $\{1,2,3,8\}$.

As Example 4.1.5 illustrates, maximality dismisses undesirable complete TRSs like empty or singletons in $\mathfrak{T}(\mathcal{C})$. This is one major source of efficiency in maximal completion. We refer to the subsequent two sections for further discussion on $\mathfrak{R}(\mathcal{C})$ and $S(\mathcal{C})$.

4.2 Automation

We describe how to automate the approach of Section 4.1.

4.2.1 Computing Maximal Terminating TRSs

Since termination is undecidable, for automation we compute maximal elements in the set of TRSs over a given $\mathcal{C}$, for which we can show termination with reduction orders automatically. However, since there are exponentially many TRSs over $\mathcal{C}$ in general, it is impractical to check termination of each of them to compute maximal elements. We present a solution using MaxSAT solving.

In the last years, SAT/SMT-encodings of termination conditions based on existing subclasses of reduction orders have been extensively investigated, and today

they are well-established. Here we mention recursive path orders [52, 17], Knuth-Bendix orders [87] and orders based on matrix interpretations [23]. Importantly, all of them can test the existence of a reduction order $>$ that satisfies arbitrary Boolean combinations of order constraints:

$$C ::= s > t \mid \top \mid \bot \mid \neg C \mid C \vee C \mid C \wedge C$$

We exploit this fact to encode a maximal termination problem into a maximal satisfiability problem. Even though NP-hard in general, nowadays solving can be efficiently done by SMT solvers.

Computing maximal terminating TRSs is done iteratively: Given a set of equalities $\mathcal{C}$, assume we already found k maximal terminating TRSs $\mathcal{R}_1, \ldots, \mathcal{R}_k$ over $\mathcal{C}$. In the following we write

$$\text{Maximize } S \text{ subject to } \varphi$$

for the problem of finding an assignment (here an order) that maximizes the number of satisfied elements in the set S while satisfying φ. We construct the following optimization problem ψ:

$$\text{Maximize } \{(s > t) \vee (t > s)\}_{s \approx t \in \mathcal{C}} \text{ subject to } \bigwedge_{i=1}^{k} \bigvee_{\ell \to r \in (\mathcal{C} \cup \mathcal{C}^{-1}) \setminus \mathcal{R}_i} \ell > r$$

Since each $\ell > r$ can be encoded w.r.t. a particular class of reduction orders to Boolean constraints, ψ can be treated as an instance of MaxSAT/MaxSMT. A solution yields a maximal subset of oriented equalities from $\mathcal{C}$, that forms a terminating TRS $\mathcal{R}_{k+1}$ and is different from all $\mathcal{R}_1, \ldots, \mathcal{R}_k$. If ψ is unsatisfiable, we found all maximal terminating TRSs over $\mathcal{C}$ (w.r.t. the considered reduction order) and return $\{\mathcal{R}_1, \ldots, \mathcal{R}_k\}$. Otherwise, we re-encode ψ w.r.t. $\mathcal{R}_{k+1}$ for another MaxSAT/MaxSMT-instance.

One might wonder, why one has to maximize all possible rules subject to

$$\bigwedge_{i=1}^{k} \bigwedge_{\ell \to r \in (\mathcal{C} \cup \mathcal{C}^{-1}) \setminus \mathcal{R}_i} \ell > r.$$

Intuitively one might think that

$$\bigwedge_{i=1}^{k} \bigvee_{\ell \to r \in \mathcal{R}_i} \neg(\ell > r)$$

is sufficient. However, note that $\neg(\ell > r)$ holds, if $\ell > r$ does not hold in the considered order, and this does not mean that $r > \ell$. While such an encoding is sound, it does generate TRSs that are not always maximal terminating. In the following example, we assume that our constraint encodings are further translated to test the existence of a lexicographic path order.

Example 4.2.1 (continued from Example 4.1.5)**.** *Recall the ES $\mathcal{E}$ consisting of:*

$$1\colon\quad \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2\colon\quad \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3\colon\quad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

We have found no maximal TRS yet, thus $k = 0$. The corresponding constraint is:

$$\textit{Maximize}\quad \{(1 \vee 1'), (2 \vee 2'), (3 \vee 3')\}\quad \textit{subject to}\quad \top$$

This constraint can be further translated into a problem over boolean constraints, that tests the existence of a suitable lexicographic path order such that all the termination constraints are satisfied. The LPO with precedence $+ \succ \mathsf{s}$ maximizes the above constraints, as it yields the TRS $\mathcal{R}_1 = \{1, 2, 3\}$. To find another maximal TRS $\mathcal{R}_2$, we encode the following constraint problem:

$$\textit{Maximize}\quad \{(1 \vee 1'), (2 \vee 2'), (3 \vee 3')\}\quad \textit{subject to}\quad (1' \vee 2' \vee 3')$$

A valid solution to this maximality constraint is the TRS $\mathcal{R}_2 = \{1, 2, 3'\}$, induced from the LPO with precedence $\mathsf{s} \succ +$. For the next constraint problem

$$\textit{Maximize}\quad \{(1 \vee 1'), (2 \vee 2'), (3 \vee 3')\}\quad \textit{subject to}\quad (1' \vee 2' \vee 3') \wedge (1' \vee 2' \vee 3)$$

the "subject to"-part is not satisfiable, so no terminating TRS can satisfy this constraint. In particular, there exists no LPO that satisfies it. Thus all maximal terminating TRSs have been computed. Now suppose we would have instead used the other encoding:

$$\textit{Maximize}\quad \{(s > t) \vee (t > s)\}_{s \approx t \in \mathcal{C}}\quad \textit{subject to}\quad \bigwedge_{i=1}^{k} \bigvee_{\ell \to r \in \mathcal{R}_i} \neg(\ell > r)$$

Then, after computing $\mathcal{R}_1$ and $\mathcal{R}_2$ similar as above, we have the following constraint.

$$\textit{Maximize}\quad \{(1 \vee 1'), (2 \vee 2'), (3 \vee 3')\}\quad \textit{subject to}\quad (\neg 1' \vee \neg 2' \vee \neg 3') \wedge (\neg 1' \vee \neg 2' \vee \neg 3)$$

If one takes the LPO with empty precedence, both sides of equation 3 are incomparable. Thus the constraint is satisfiable, and we obtain the TRS $\mathcal{R} = \{1, 2\}$. This TRS is indeed terminating, but it is not maximal terminating.

Finally, in our implementation we do not compute all maximal terminating TRSs. This is because there still may be exponentially many maximal terminating TRSs. Instead, we fix a number K to stop the enumeration of maximal terminating TRSs whenever the number reaches K. This is motivated by the following observation: Assume that there exists a complete TRS $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, but we fail to select it. Since $\mathcal{R}$ is a terminating TRS over $\mathcal{C} \cup S(\mathcal{C})$, by Lemma 4.1.4 (a) there exists a maximal terminating, complete TRS $\mathcal{R}' \in \mathfrak{R}(\mathcal{C} \cup S(\mathcal{C}))$ with $\mathcal{R} \subseteq \mathcal{R}'$. Thus when missing the complete TRS $\mathcal{R}$ in one iteration, there is still a chance to select $\mathcal{R}'$ in the next one.

4.2.2 Filtering Equalities

Our implementation of the parameter function $S(\mathcal{C})$ follows closely the proposed one of Section 4.1, but adds a few small operations as described below.

When orienting equalities to rules, some equalities tend to generate a lot of critical pairs. This is why Knuth-Bendix completion employs selection heuristics (cf. [10, 82, 84]) that select only certain kinds of equalities. We also heuristically select equalities, since otherwise the number of critical pairs grows too fast and our implementation fails to handle it. In order to address it, we first normalize the equalities to filter out all those whose size exceeds a bound d. Then, we select the n smallest equalities. If they are not uniquely defined, i.e. there are several equalities of the same size, we simply select equalities in the order in which they were generated. We formulate this filtering. For a set of equalities $\mathcal{C}$, we write $\mathcal{C}^{<d}$ to denote all equalities $s \approx t$ of $\mathcal{C}$ with $|s| + |t| < d$. Moreover we write $\mathcal{C}\restriction_n$ for the set of the n smallest equalities in $\mathcal{C}$. With these notations, $S(\mathcal{C})$ can be described as follows:

$$S(\mathcal{C}) = \bigcup_{\mathcal{R} \in \mathfrak{R}(\mathcal{C})} \left(\{ s{\downarrow_{\mathcal{R}}} \approx t{\downarrow_{\mathcal{R}}} \mid s \approx t \in \mathcal{E} \cup \mathsf{CP}(\mathcal{R}) \text{ and } s{\downarrow_{\mathcal{R}}} \neq t{\downarrow_{\mathcal{R}}} \}^{<d} \restriction_n \right)$$

4.3 Related Work

We relate our procedure φ to existing completion methods described in Chapter 3. Due to their algorithmic nature precise simulations are difficult, but we capture their main features. We say that $\mathcal{S}$ is an *inter-reduced* version of a terminating TRS $\mathcal{R}$, if $\mathcal{S}$ is a terminating reduced TRS whose rules are obtained by rewriting rules in $\mathcal{R}$ by $\mathcal{R}$ itself.

- **Knuth-Bendix Completion**. Let $>$ be a reduction order for the Knuth-Bendix completion procedure and for the orientable part $\{\ell \to r \in \mathcal{C} \cup \mathcal{C}^{-1} \mid \ell > r\}$ we write $\mathcal{C}^{>}$. The Knuth-Bendix completion procedure (cf. Section 3.1) can be simulated by φ if one uses

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{C}^{>}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{C}^{>}$ and $s \approx t \in \mathcal{C} \cup \mathsf{CP}(\mathcal{R}')$.

- **Multi-completion**. As mentioned in Section 3.3, Multi-completion uses a class of reduction orders $>_1, \ldots, >_n$ to run Knuth-Bendix completion in parallel, and typically the class is the set of all possible recursive path orders. Its run can be simulated in our method as follows:

  $$\mathfrak{R}(\mathcal{C}) = \{\mathcal{C}^{>_1}, \ldots, \mathcal{C}^{>_n}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

where, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{C}^{>_i}$ and $s \approx t \in \mathcal{C} \cup \mathsf{CP}(\mathcal{R}')$. A naive implementation of this approach fails due to the large number of compatibility checks as well as computations of normal forms. As illustrated in Chapter 3, the specialised data structure *node* allows to share these computations among the orders.

- **Completion with termination tools**. As introduced in Section 3.2, this procedure does not require a reduction order as an input parameter, because during its process a necessary reduction order is constructed on the fly:

$$\mathfrak{R}(\mathcal{C}) = \{\mathcal{R}\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}$ is a TRS over $\mathcal{C}$ whose termination is shown by a *termination tool*, $\mathcal{R}'$ is an inter-reduced version of $\mathcal{R}$, and $s \approx t \in \mathcal{C} \cup \mathsf{CP}(\mathcal{R}')$. Unlike a fixed single reduction order, a termination tool can find a number of terminating TRSs over $\mathcal{C}$, which avoids failure of Knuth-Bendix completion. The main drawback of this procedure is similar as with multi-completion, and in n [82] a heuristic for the *best-first search strategy* is suggested to select one of the terminating TRSs. Nevertheless, this approach significantly extends the power of Knuth-Bendix completion, and has been adopted in their completion tool Slothrop.

- **Multi-completion with termination tools** (cf. Chapter 3.4). The method replaces reduction orders in multi-completion by a termination tool:

$$\mathfrak{R}(\mathcal{C}) = \{\mathcal{R}_1, \ldots, \mathcal{R}_n\} \qquad \text{and} \qquad S(\mathcal{C}) = \{s{\downarrow_{\mathcal{R}'}} \approx t{\downarrow_{\mathcal{R}'}}\}$$

  where, $\mathcal{R}_1, \ldots, \mathcal{R}_n$ are all TRS over $\mathcal{C}$ whose termination is shown by a termination tool, $\mathcal{R}'$ is an inter-reduced version of some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, and $s \approx t \in \mathcal{C} \cup \mathsf{CP}(\mathcal{R}')$. A variant of the node data structure that is used in multi-completion provides a compact representation of $\mathfrak{R}(\mathcal{C})$ as well as an efficient algorithm to compute it. This approach has been implemented in the very effective completion tool mkbTT.

As stated before, maximal completion only computes maximal terminating TRSs, which are often much fewer than all terminating TRSs, but it does not miss a complete TRS. This is the main idea of our approach. One drawback is the current limited power of maximal termination provers. Theoretically, brute force search allows using a termination tool to compute maximal terminating TRSs. However, it is practically infeasible due to exponentially many calls of the termination tool.

Another difference is the definition of $S(\mathcal{C})$. Except for maximal completion, all procedures use a single equality in $S(\mathcal{C})$ and its *selection* is critical for successful runs. The next example illustrates this.

Table 4.1: Summary for all 115 equational systems

	LPO		KBO		termination tool	
	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
completed	70	**86**	67	**69**	81	71
failure	6	**6**	3	**3**	3	4
timeout	39	**23**	45	**43**	31	40

Example 4.3.1. *Recall the ES $\mathcal{E}$ in Example 4.1.5:*

$$1\colon\quad \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2\colon\quad \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3\colon\quad \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

We perform $\varphi(\mathcal{E})$ as the simulated run of multi-completion, where the class of reduction orders is all LPOs with total precedence. Assume that our selection strategy for $S(\mathcal{C})$ prefers an equality derived from the critical pair of rule 3 and the rule of the biggest possible index for some TRS in $\mathfrak{R}(\mathcal{C})$.

(i) $\mathfrak{R}(\{1,2,3\}) = \{\{1,2,3\},\{1,2,3'\}\}$*, which both do not satisfy the join-condition of φ. Thus, $\varphi(\{1,2,3\}) = \varphi(\{1,\dots,4\})$, where 4 is the single equality in $S(\{1,2,3\})$:*

$$4\colon\quad x + y \approx \mathsf{s}(\mathsf{p}(x) + y)$$

(ii) $\mathfrak{R}(\{1,\dots,4\}) = \{\{1,2,3,4'\},\{1,2,3',4'\}\}$ *and the join-condition does not hold again. We continue the run with $\varphi(\{1,\dots,4\}) = \varphi(\{1,\dots,5\})$, where 5 in $S(\{1,\dots,4\})$ is*

$$5\colon\quad (x + y) + z \approx \mathsf{s}((\mathsf{p}(x) + y) + z)$$

(iii) Generally, we have $\mathfrak{R}(\{1,\dots,n\}) = \{\{1,2,3,4',\dots,n'\},\{1,2,3',4',\dots,n'\}\}$ and $S(\{1,\dots,n\})$ is the singleton of

$$n{+}1\colon\quad ((x_1 + x_2) + \cdots) + x_{n-1} \approx \mathsf{s}(((\mathsf{p}(x_1) + x_2) + \cdots) + x_{n-1})$$

for $n \geq 3$. Thus, the join-condition never holds and the procedure does not terminate.

Admittedly, for the above example it is easy to choose an appropriate selection strategy that succeeds. In general however, it is difficult to know a suitable selection strategy a priori. This is why mkbTT provides several selection strategies

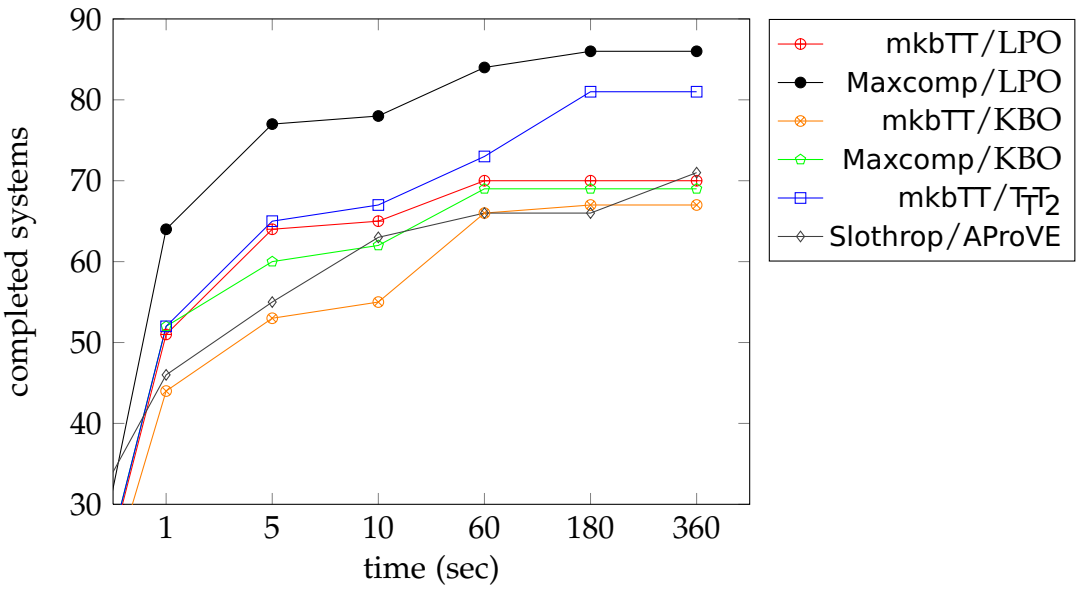

Figure 4.1: Comparison of execution times.

as a user parameter. Maximal completion does not use a singleton but a *set* of equalities for $S(\mathcal{C})$, which reduces the risk to get stuck.

To conclude, we like to stress the simplicity of maximal completion, due to avoiding a dedicated search algorithm like the one in Slothrop, and a sophisticated but complex data structure like that of multi-completion. The only major drawback is that maximal completion is practically limited to methods of showing termination that can be encoded to constraints. This can prevent to find a complete TRS even if one exists for a given ES. For example to complete the theory CGE_n of n commuting group endomorphisms [82], advanced termination techniques are required which can not be encoded as a constraint problem. Consequently, maximal completion cannot (practically) complete CGE_n.

4.4 Experiments

We implemented maximal completion in the tool Maxcomp. The tool employs the SMT solver Yices [22] to support LPO and KBO. Concerning parameter K for $\mathfrak{R}(\mathcal{C})$ in Section 4.2, at the beginning we use $K = 2$ to compute two maximal terminating TRSs. During the recursion of φ, we increase K whenever $S(\mathcal{C}) \subseteq \mathcal{C}$. If, at some point, no new equalities are generated and all maximal terminating TRSs are computed (i.e. the parameter K cannot be increased any more), the tool stops with failure. Moreover, we fix $n = 7$ and $d = 20$ for $S(\mathcal{C})$. These values were chosen heuristically, see below for a brief discussion on their effect.

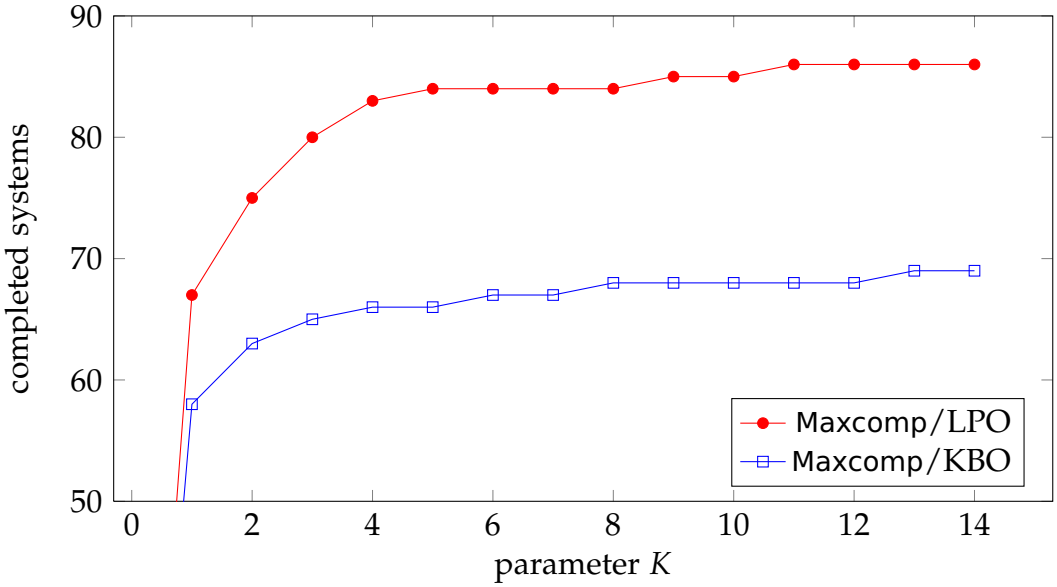

Figure 4.2: Computation of maximal terminating TRSs ("*K*").

We compare Maxcomp with the two existing completion tools Slothrop and mkbTT. Since the latter two require termination provers, we used AProVE [28] for Slothrop, and for mkbTT its internally supplied prover TTT2 [50]. The test-bed consists of 115 equational systems from the distribution of mkbTT.[1] The tests were single-threaded run on a system equipped with an Intel Core Duo L7500 with 1.6 GHz and 2 GB of RAM using a timeout of 300 seconds.

Table 4.1 gives a summary of the overall results. Here we also included results, where mkbTT's termination proving power was limited to LPO and KBO[2] (this is not possible for Slothrop). Aside from this, all parameters of all tools were left at their default values. A full list of all results can be found in Appendix A.

It should be noted that the complete systems found by using a termination tool are mostly different from those found with LPO or KBO, since the reduction order constructed using a termination tool is usually different from them. Table 4.1 suggests that Maxcomp succeeds whenever mkbTT or Slothrop succeed, but this is not the case. In fact for each tool there exist equational systems for which the tool succeeds to find a complete TRS, but both of the other tools fail to do so.

For almost all equational systems in the test-set complete TRSs are known. The reason why the tools fail or time out however is different with respect to each tool. Maxcomp mostly fails or times out when it is not possible to show termination of the known complete TRS for the ES in question by LPO or KBO. For Slothrop, the reason for a timeout is usually that its best-first search algorithm picks the

[1] `http://cl-informatik.uibk.ac.at/software/mkbtt/`
[2] `mkbtt -s lpo` for LPO, and `mkbtt -s kbo` for KBO.

wrong branch in the search tree and is unable to recover from that, and mkbTT faces a similar problem. It should be noted however that by choosing specific, suitable selection strategies for each equational system individually, mkbTT can complete more systems than with its default selection strategy. To name one example, mkbTT completes SK90.3.22 with LPO and selection strategy old [84] within roughly 40 seconds, however times out after 300 seconds with all other predefined strategies (max, slothrop, sum). While the chosen selection strategy vastly affects the outcome of mkbTT, it is in general non-trivial to decide which selection strategy to choose in advance.

Concerning the timeout, with very few exceptions, a higher timeout seems not to affect the results. A notable exception are the group axioms with two commuting endomorphisms CGE2 [82], for which Slothrop was the first tool to automatically construct a complete TRS. Here however Slothrop fails to complete CGE2 within 300 seconds.

All in all, as can be seen in Figure 4.1, Maxcomp is the fastest tool. The reason is, that significant time during completion is spent on termination analysis, and the constraint-based approach of Maxcomp outperforms the use of termination tools. For overall performance, whenever all tools succeeded, they usually (with four exceptions) did so in less than 35 seconds. For the rest of systems, timing values do not show a clear trend. As seen in Figure 4.2, the parameter K mostly remains unchanged at 2, and for the vast majority of successful runs does not exceed 5, the maximum being 14. It seems that the computation of few maximal terminating TRSs suffices to guide the search process in the right direction.

One might think that other parameters (n and k) of Maxcomp should heavily affect the outcome of the experiments, but in practice their overall influence is rather small. For example with settings $n = 7$ and $d = 21$, and $n = 7$ and $d = 19$ using LPO, the completed systems did not change at all. For $n = 8$ and $d = 20$ the TRS LS94.P1.trs could be completed, but completion of SK90.3.28.trs did not terminate. Lastly for $n = 6$ and $d = 20$ the TRSs LS94.P1.trs and SK90.3.26.trs were completed, but again, the run on the ES SK90.3.28.trs did not terminate. An explanation for this behaviour is that with no bound n and k at all we have exponential growth of the set $\mathcal{C}$ due to the exponential number of critical pairs. However, fixing n and k with any values ensures that the growth of $\mathcal{C}$ becomes constant, which suffices to get practically good results.

Chapter 5

Constrained Equalities

Most calculi for (inductive) theorem proving are based on the idea of finding a terminating TRS that satisfies certain joinability requirements of equalities (cf. Chapter 3). Thereby an equational proof is transformed into a search problem. To find such a TRS the inference rules of completion are adapted. However they usually require a fixed reduction ordering as an input parameter.

Chapter 4 illustrated how completion can be fully automated by reformulating it as an optimization problem over constraints. As seen, this also yields a practically highly efficient procedure. The key ingredient in the reformulation is the use of *maximality* (Lemma 4.1.4). It states that given a set of candidate equalities, it can never be harmful to choose a terminating superset of the required rules. Thus one can maximize the choice of rules with respect to orientability.

In inductive theorem proving, such a choice does not fit, even if only (inductively) valid equations are considered. For example, in rewriting induction an analogous formulation would be that if $\mathcal{R}$ and $\mathcal{H}$ are TRSs with $\mathcal{R}$ quasi-reducible and $\mathcal{R} \cup \mathcal{H}$ terminating and $\mathrm{Expd}(\mathcal{R}, \mathcal{H}) \subseteq {\downarrow_{\mathcal{R}\cup\mathcal{H}}}$, then for all $\mathcal{H}'$ with $\mathcal{H} \subseteq \mathcal{H}'$ and $\mathcal{R} \cup \mathcal{H}'$ terminating and $\mathcal{R} \vdash_{\mathrm{i}} \mathcal{H}'$ we have $\mathrm{Expd}(\mathcal{R}, \mathcal{H}') \subseteq {\downarrow_{\mathcal{R}\cup\mathcal{H}'}}$, cf. Theorem 3.5.3. Unfortunately this is not the case.

Example 5.0.1. *Recall the TRS $\mathcal{R}$ of Example 3.5.5:*

$$1\colon \quad 0 + y \to y \qquad\qquad 2\colon \quad \mathsf{s}(x) + y \to \mathsf{s}(x + y)$$

Suppose we have additionally the following two equations

$$3\colon \quad (x+y)+z \approx (x+y)+z \qquad\qquad 4\colon \quad x+\mathsf{s}(x) \approx \mathsf{s}(x+x)$$

and we want to prove that $\mathcal{R} \vdash_i \{3\}$. *One can verify that indeed all the conditions of the above (false) conjecture are met. Suppose we orient rules* 3 *and* 4 *from left to right and expand at leftmost, innermost basic positions. If we take* $\mathcal{H} = \{3\}$ *then* $\mathrm{Expd}(\mathcal{R},\mathcal{H}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{H}}$ *which implies* $\mathcal{R} \vdash_i \{3\}$ *by Theorem 3.5.3. However when choosing the superset* $\mathcal{H} = \{3,4\}$ *we have* $\mathrm{Expd}(\mathcal{R},\mathcal{H}) \not\subseteq \downarrow_{\mathcal{R}\cup\mathcal{H}}$ *even though both* 3 *and* 4 *are valid inductive theorems.*

To summarize, choosing an appropriate set of equations $\mathcal{H}$ among candidates to solve the associated joinability problem $\mathrm{Expd}(\mathcal{R},\mathcal{H}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{H}}$ is a highly non-trivial task. Clearly, maximality with respect to the number of orientations is not a suitable criterion.

To address this problem we attach constraints to each equation. From such *constrained equalities* one can infer information on how selecting this equation will affect the joinability problem, namely how many unjoinable equations it will induce. Such additional information allow to again formulate these more complicated joinability requirements as optimization problems, and automate them in a similar fashion to maximal completion.

The framework turns out to be very flexible. Due to the abstract nature of the constraints it allows easy adaptations to various settings. In Section 5.1 we introduce the framework of constrained equalities, and in Section 5.2 we show how inductionless induction and rewriting induction (cf. Chapter 3) can be formulated in it. In maximal completion there is no possibility to apply *inter-reduction*. In contrast to maximal completion, constrained equalities allow to formulate completion with inter-reduction as well, which is also done in Section 5.2, and in Section 5.3 it is shown how such an approach can be automated. In the practical experiments of Section 5.4 however it turns out that exploiting maximality in completion and maximizing as in maximal completion is more effective than using an approach based on constrained equalities. In contrast, for our main objective, namely inductive methods, we obtain a fully automatic and practically efficient procedure. Finally we compare our approach with related work in Section 5.5.

5.1 Constrained Equalities

We introduce *termination constraints*. The syntax is given by the next BNF C

$$C ::= \ell \to r \mid \top \mid \bot \mid \neg C \mid C \vee C \mid C \wedge C$$

A TRS $\mathcal{R}$ *satisfies* a constraint C if $\mathcal{R} \models C$ holds. Here $\mathcal{R} \models C$ is inductively defined on C as the standard interpretation of Boolean connectives together with the following base case: $\mathcal{R} \models \ell \to r$ if and only if $\ell \to r \in \mathcal{R}$.

A *constrained equality* is a pair of an equality on terms and a constraint. A *constrained equational system*, in short CES, $\mathcal{C}$ is a set of constrained equalities. Let $\mathcal{R}$ be a terminating TRS. The $\mathcal{R}$-projection of $\mathcal{C}$ is the ES given by

$$\{s \approx t \mid (s \approx t, C) \in \mathcal{C} \text{ and } \mathcal{R} \models C\}$$

and is denoted by $\mathcal{C}[\![\mathcal{R}]\!]$. For brevity, we define the following notations:

$$\begin{aligned}
\mathcal{E}^{\top} &= \{(s \approx t, \top) \mid s \approx t \in \mathcal{E}\} \\
\mathcal{C} \ominus \mathcal{R} &= \left\{ \left(s \approx t, C \wedge \neg \bigwedge \mathcal{R}\right) \mid (s \approx t, C) \in \mathcal{C} \right\} \\
\mathcal{C}{\downarrow_{\mathcal{R}}} &= \left\{ \left(s{\downarrow_{\mathcal{R}}} \approx t{\downarrow_{\mathcal{R}}}, C \wedge \bigwedge \mathcal{R}\right) \mid (s \approx t, C) \in \mathcal{C} \text{ and } s{\downarrow_{\mathcal{R}}} \neq t{\downarrow_{\mathcal{R}}} \right\}
\end{aligned}$$

The intuition behind these notations is as follows. $\mathcal{E}^{\top}$ denotes constrained equalities obtained from sets of equalities, and $\mathcal{C}{\downarrow_{\mathcal{R}}}$ rewrites the equational part in $\mathcal{C}$ to some normal form. However an added constraint ensures that the newly obtained normal forms will be only triggered by the TRS $\mathcal{R}$ with which they were normalized with. Lastly $\mathcal{C} \ominus \mathcal{R}$ removes $\mathcal{R}$ from the constrained part, and thus we have $\mathcal{R} \not\models \mathcal{C} \ominus \mathcal{R}$. The next lemmata state important properties of these notations. Their proofs are straightforward by unfolding the above definitions.

Lemma 5.1.1. *If* $\mathcal{E}^{\top}{\downarrow_{\mathcal{R}}} = \varnothing$ *then* $\mathcal{E} \subseteq {\downarrow_{\mathcal{R}}}$. ∎

Lemma 5.1.2. $(\mathcal{C} \ominus \mathcal{R})[\![\mathcal{R}]\!] = \varnothing$. ∎

Our main concern is a joinability problem of $\mathcal{C}$, namely, $\mathcal{C}[\![\mathcal{R}]\!] \subseteq {\downarrow_{\mathcal{R}}}$. But for inductive theorem proving, one has only to consider ground joinability: We say that an ES $\mathcal{E}$ is *ground joinable for* $\mathcal{R}$ if $s\sigma \downarrow_{\mathcal{R}} t\sigma$ whenever $s \approx t \in \mathcal{E}$ and $s\sigma, t\sigma \in \mathcal{T}(\mathcal{F})$. A function S is a *(ground) reduction* for (ground) joinability problems if for every constrained ES $\mathcal{C}$ and every TRS $\mathcal{R}$ (ground) joinability of $S(\mathcal{C})[\![\mathcal{R}]\!]$ for $\mathcal{R}$ implies that of $\mathcal{C}[\![\mathcal{R}]\!]$. We give a simple criterion for testing reductionism of S. Below we say $A \subseteq B$ *on (ground) terms*, if for all $s \approx t \in A$ and all (ground) substitutions σ on s and t we have $s\sigma \approx t\sigma \in B$.

Lemma 5.1.3. *S is a (ground) reduction if* $\mathcal{C}[\![\mathcal{H}]\!] \subseteq {\to_{\mathcal{H}}^{*}} \cdot {\leftrightarrow_{S(\mathcal{C})[\![\mathcal{H}]\!]}^{=}} \cdot {\leftarrow_{\mathcal{H}}^{*}}$ *on (ground) terms for all $\mathcal{C}$ and $\mathcal{H}$.*

Proof. We show that S is a (ground) reduction if the inclusion of the lemma holds for (ground) terms. Let $\mathcal{C}$ be a CES and $\mathcal{H}$ be a TRS. Assume $S(\mathcal{C})[\![\mathcal{H}]\!]$ is (ground) joinable for $\mathcal{H}$. Then for any $s \approx t \in S(\mathcal{C})[\![\mathcal{H}]\!]$ and any substitution σ we have $s\sigma \downarrow_{\mathcal{H}} t\sigma$. Together with

$$\mathcal{C}[\![\mathcal{H}]\!] \subseteq {\to_{\mathcal{H}}^{*}} \cdot {\leftrightarrow_{S(\mathcal{C})[\![\mathcal{H}]\!]}^{=}} \cdot {\leftarrow_{\mathcal{H}}^{*}}$$

we have $\mathcal{C}[\![\mathcal{H}]\!] \subseteq \downarrow_{\mathcal{H}}$. Thus (ground) joinability of $\mathcal{C}[\![\mathcal{H}]\!]$ for $\mathcal{H}$ holds. Hence S is a reduction. ∎

5.2 Constraints for Joinability Problems

In this section we show how completion and variants can be recast as joinability problems based using reductions.

Definition 5.2.1. *Let $\mathfrak{R}$ be a function from constrained ESs to a set of terminating TRSs, and F be a function from TRSs to ESs. In the following we use the notation*

$$S_{\mathcal{R}}(\mathcal{C}) = (\mathcal{C} \ominus \mathcal{R}) \cup \mathcal{C}{\downarrow_{\mathcal{R}}} \cup F(\mathcal{R})^{\top}{\downarrow_{\mathcal{R}}}, \text{ and}$$
$$S(\mathcal{C}) = \bigcup_{\mathcal{R} \in \mathfrak{R}(\mathcal{C})} S_{\mathcal{R}}(\mathcal{C}).$$

Lemma 5.2.2. *S is a (ground) reduction.*

Proof. Let $s \approx t \in \mathcal{C}[\![\mathcal{H}]\!]$ and $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$. According to Lemma 5.1.3 it is sufficient to show $s\sigma \to_{\mathcal{H}}^{*} \cdot \overline{\leftrightarrow}_{S(\mathcal{C})[\![\mathcal{H}]\!]} \cdot {}_{\mathcal{H}}^{*}\!\leftarrow t\sigma$ for all (ground) substitutions σ. By the assumption there is a C with $(s \approx t, C) \in \mathcal{C}$ and $\mathcal{H} \models C$. We distinguish three cases.

- If $\mathcal{R} \subseteq \mathcal{H}$ and $s{\downarrow_{\mathcal{R}}} = t{\downarrow_{\mathcal{R}}}$ then $s\sigma \downarrow_{\mathcal{R}} t\sigma$ for all (ground) substitutions σ, and thus $s\sigma \downarrow_{\mathcal{H}} t\sigma$.
- If $\mathcal{R} \subseteq \mathcal{H}$ and $s{\downarrow_{\mathcal{R}}} \neq t{\downarrow_{\mathcal{R}}}$ then $\mathcal{H} \models C \wedge \bigwedge \mathcal{R}$. Thus $s{\downarrow_{\mathcal{R}}} \approx t{\downarrow_{\mathcal{R}}} \in S_{\mathcal{R}}(\mathcal{C})[\![\mathcal{H}]\!]$. Therefore $s\sigma \to_{\mathcal{H}}^{*} s{\downarrow_{\mathcal{R}}} \leftrightarrow_{S_{\mathcal{R}}(\mathcal{C})[\![\mathcal{H}]\!]} t{\downarrow_{\mathcal{R}}} {}_{\mathcal{H}}^{*}\!\leftarrow t\sigma$ for all (ground) substitutions σ.
- If $\mathcal{R} \not\subseteq \mathcal{H}$ then $\mathcal{H} \models C \wedge \neg \bigwedge \mathcal{R}$. Thus, we have $s \approx t \in S_{\mathcal{R}}(\mathcal{C})[\![\mathcal{H}]\!]$, which implies $s\sigma \leftrightarrow_{S_{\mathcal{R}}(\mathcal{C})[\![\mathcal{H}]\!]} t\sigma$ for all (ground) substitutions σ.

∎

Lemma 5.2.3. *If $S(\mathcal{C})[\![\mathcal{R}]\!] = \varnothing$ and $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, then $F(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$.*

Proof. Let $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$ and $s \approx t \in F(\mathcal{R})$. Because $\mathcal{R} \models \bigwedge \mathcal{R}$ and $S(\mathcal{C})[\![\mathcal{R}]\!] = \varnothing$, we must have $s{\downarrow_{\mathcal{R}}} = t{\downarrow_{\mathcal{R}}}$, and thus $s \downarrow_{\mathcal{R}} t$. ∎

5.2.1 Completion

In this section, we specify parameters to derive our completion criterion with *inter-reduction*. Let $\mathcal{E}$ be an ES. Below we assume that $S_{KB}(\mathcal{C})$ is an instance of the above reduction S so that $\mathfrak{R}(\mathcal{C})$ is a set of terminating TRSs with $\mathcal{R} \subseteq \leftrightarrow_{\mathcal{E}}^{*}$ for all $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, and $F(\mathcal{R}) = \mathsf{CP}(\mathcal{R})$.

Theorem 5.2.4. *Let $\mathcal{C} = S^n_{KB}(\mathcal{E}^\top)$. If $\mathcal{C}[\![\mathcal{R}]\!] = \varnothing$ for some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, then $\mathcal{R}$ is complete for $\mathcal{E}$.*

Proof. Let $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$ and $\mathcal{C}[\![\mathcal{R}]\!] = \varnothing$. Easy induction on n shows that the n-fold composition of a reduction forms a reduction. Thus, reductionism of S^n_{KB} is implied by Lemma 5.2.2 and we obtain $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$. We distinguish two cases: If $n = 0$ then $\mathcal{E}$ must be empty. Since termination of $\mathcal{R}$ and the inclusion $\mathcal{R} \subseteq \leftrightarrow^*_{\mathcal{E}}$ entail emptiness of $\mathcal{R}$, the claim trivially holds. If $n > 0$ then $S_{KB}(\mathcal{C})[\![\mathcal{R}]\!] = \varnothing$ holds. Since $\mathcal{R} \subseteq \leftrightarrow^*_{\mathcal{E}}$ holds by assumption we have $\leftrightarrow^*_{\mathcal{R}} \subseteq \leftrightarrow^*_{\mathcal{E}}$. Moreover, $\mathcal{R}$ is terminating by construction and confluent by Lemma 5.2.3. Thus $\mathcal{R}$ is complete for $\mathcal{E}$. ∎

In the following examples we assume that $\mathfrak{R}(\mathcal{C})$ is a singleton.

Example 5.2.5. *Consider the ES $\mathcal{E}$ consisting of the following equations:*

$$1 : \mathsf{s}(\mathsf{p}(x)) \approx x \qquad 2 : \mathsf{p}(\mathsf{s}(x)) \approx x \qquad 3 : \mathsf{s}(x) + y \approx \mathsf{s}(x + y)$$

We thus obtain three initial constraints $\mathcal{C}_0 = \mathcal{E}^\top = \{(1, \top), (2, \top), (3, \top)\}$. Assuming $\mathcal{R}_0 = \{1, 2, 3\} \in \mathfrak{R}(\mathcal{C}_0)$ and abbreviating $R_0 = 1 \wedge 2 \wedge 3$ results in the constraints $\mathcal{C}_1 = \{(1, \neg R_0), (2, \neg R_0), (3, \neg R_0), (4, \top)\}$ where 4 is derived from the critical overlap $\langle 1, 1, 3\rangle$:

$$4 : \quad x + y \approx \mathsf{s}(\mathsf{p}(x) + y)$$

Now we can for instance take $\mathcal{R}_1 = \{1, 2, 3', 4'\} \in \mathfrak{R}(\mathcal{C}_1)$. We write R_1 as a shorthand for $1 \wedge 2 \wedge 3' \wedge 4'$, and obtain

$$\mathcal{C}_2 = \{(1, \neg R_0 \wedge \neg R_1), (2, \neg R_0 \wedge \neg R_1), (3, \neg R_0 \wedge \neg R_1), (4, \neg R_1), (5, \top), (6, \top)\}$$

where 5 and 6 are new constrained equalities obtained from the critical overlaps $\langle 3', 1, 2\rangle$ and $\langle 4', 1, 2\rangle$, respectively:

$$5 : \quad (\mathsf{p}(\mathsf{s}(x) + y) \approx x + y, \top) \qquad 6 : \quad (\mathsf{p}(x + y) \approx \mathsf{p}(x) + y, \top)$$

For $\mathcal{R}_2 = \mathcal{R}_0 \cup \{4', 5, 6'\} \in \mathfrak{R}(\mathcal{C}_2)$ we now have $S_{KB}(\mathcal{C}_2)[\![\mathcal{R}_2]\!] = \varnothing$, so $\mathcal{R}_2$ is complete for $\mathcal{E}$. Note that this would also hold for the smaller (inter-reduced) system $\mathcal{R}'_2 = \{1, 2, 3, 6'\}$, which is actually complete.

We remark that in Example 5.2.5 $S_{KB}(\mathcal{C}_2)[\![\mathcal{R}'_2]\!] = \varnothing$, and thus completeness of $\mathcal{R}'_2$ can be inferred directly from the attached constraints in $\mathcal{C}_2$. In maximal completion this information is not available, and a non-inter-reduced TRS such as $\mathcal{R}_2$ is always selected (cf. Example 4.1.5). Of course one question is how to automatically construct $\mathcal{R}'_2$ from $\mathcal{C}_2$. This is subject to Section 5.3.

5.2.2 Inductionless Induction

Furthermore, Gramlich's criterion 3.5.2 for inductionless induction can be instantiated. Note that since talking about inductive theorem proving, throughout this and the next section we consider TRSs to be sorted (cf. Chapter 3).

We recall Theorem 3.5.2:

Lemma 5.2.6 ([34]). *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{H}$ be a terminating, and left-$\mathcal{R}_0$-inductively-reducible TRS. If $\mathsf{CP}(\mathcal{R}_0, \mathcal{H}) \subseteq \downarrow_{\mathcal{R}}$, then $\mathcal{R}_0 \vdash_i \mathcal{H}$.* ∎

Lemma 5.2.7. *Let $\mathcal{R} = \mathcal{R}_0 \cup \mathcal{H}$ be a TRS, such that $\mathcal{R}_0 \vdash_i \mathcal{H}$, and $\mathcal{E}$ be a set of equalities. If $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$ on ground terms, then $\mathcal{R}_0 \vdash_i \mathcal{E}$.* ∎

Let $\mathcal{R}_0$ be a TRS and $\mathcal{E}$ an ES. We assume that $S_G(\mathcal{C})$ is an instance of S so that $F(\mathcal{R}) = \mathsf{CP}(\mathcal{R}_0, \mathcal{R} \setminus \mathcal{R}_0)$ and $\mathfrak{R}(\mathcal{C})$ is a set of terminating and left-$\mathcal{R}_0$-inductively-reducible TRSs.

Theorem 5.2.8. *If $S_G^n(\mathcal{E}^\top)[\![\mathcal{R}]\!] = \varnothing$ for some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, then $\mathcal{R}_0 \vdash_i \mathcal{E}$.*

Proof. Let $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$ and $\mathcal{C}[\![\mathcal{R}]\!] = \varnothing$. Analogous to Theorem 5.2.4, we obtain $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$ on ground terms. Again, we distinguish two cases: If $n = 0$ then $\mathcal{E}$ must be empty, and the claim trivially holds. If $n > 0$ then $S_G(\mathcal{C})[\![\mathcal{R}]\!] = \varnothing$ holds. By Lemma 5.2.3, we have $\mathsf{CP}(\mathcal{R}_0, \mathcal{R} \setminus \mathcal{R}_0) \subseteq \downarrow_{\mathcal{R}}$. Since $\mathcal{R}$ is terminating and left-$\mathcal{R}_0$-inductively-reducible, by Lemma 5.2.6 we have $\mathcal{R}_0 \vdash_{\mathrm{i}} (\mathcal{R} \setminus \mathcal{R}_0)$. Together with $\mathcal{E} \subseteq \downarrow_{\mathcal{R}}$, by Lemma 5.2.7, we have $\mathcal{R}_0 \vdash_{\mathrm{i}} \mathcal{E}$. ∎

Example 5.2.9. *Consider the TRS $\mathcal{R}_0$ consisting of the following rules:*

$$1: \quad 0 - x \to 0 \qquad 2: \quad \mathsf{s}(x) - 0 \to \mathsf{s}(x) \qquad 3: \quad \mathsf{s}(x) - \mathsf{s}(y) \to x - y$$
$$4: \quad \mathsf{p}(0) \to 0 \qquad 5: \quad \mathsf{p}(\mathsf{s}(x)) \to x$$

and $\mathcal{E}$ the single equality

$$6: \quad \mathsf{p}(x - y) \approx \mathsf{p}(x) - y$$

Here we assume the signature to be sorted with $0 : \mathrm{nat}, \mathsf{p}, \mathsf{s} : \mathrm{nat} \to \mathrm{nat}$ and $+, - : \mathrm{nat} \times \mathrm{nat} \to \mathrm{nat}$. We obtain the initial constraints $\mathcal{C}_0 = \{(6, \top)\}$. Choosing $\mathcal{R}_1 = \{1, \ldots, 6\}$ from $\mathfrak{R}(\mathcal{C}_0)$ results in the following constraint equations

$$\mathcal{C}_1 = S_G(\mathcal{C}_0) = \{(6, \neg\mathcal{R}_1), (7, \top), (8, \top)\}$$

where $7, 8$ are

$$7: \quad x - 0 \approx x \qquad\qquad 8: \quad \mathsf{p}(x) - y \approx x - \mathsf{s}(y)$$

Let $\mathcal{R}_2 = \{1, \ldots, 8\}$ be in $\mathfrak{R}(\mathcal{C}_1)$, resulting in

$$\mathcal{C}_2 = \{(6, \neg\mathcal{R}_1 \wedge \neg\mathcal{R}_2), (7, \neg\mathcal{R}_2), (8, \neg\mathcal{R}_2)\}$$

Suppose $\mathcal{R}_2$ is again included in $\mathfrak{R}(\mathcal{C}_2)$ and chosen as $\mathcal{R}_3$. Then $S_G(\mathcal{C}_2)[\![\mathcal{R}_3]\!] = \varnothing$. One can verify that indeed $\mathcal{R}_1, \ldots, \mathcal{R}_3$ are left-$\mathcal{R}_0$-inductively reducible and terminating. Thus $\mathcal{R}_0 \vdash_i \mathcal{E}$.

5.2.3 Rewriting Induction

As mentioned in Chapter 3, Rewriting Induction [62] relaxes the joinability conditions of inductionless induction. Instead of requiring all critical pairs to be joinable, joinability of only a specific subset suffices.

In the following, we assume $\mathfrak{R}(\mathcal{C})$ to be a set of $\mathcal{R}_0$-expandable, terminating TRSs such that for every $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, we have $\mathcal{R}_0 \subseteq \mathcal{R}$ and $\ell\sigma_g \leftrightarrow^*_{\mathcal{R}_0 \cup \mathcal{E}} r\sigma_g$ for all $\ell \to r \in \mathcal{R}$ and ground substitutions σ_g. We assume $\mathcal{S}_{\mathrm{RI}}(\mathcal{C})$ to be an instance of $\mathcal{S}$ with $F(\mathcal{R}) = \mathrm{Expd}(\mathcal{R}_0, \mathcal{R})$.

Lemma 5.2.10 ([62])**.** *Let $\mathcal{R}$ and $\mathcal{H}$ be TRSs such that $\mathcal{R}$ is quasi-reducible TRS, and $\mathcal{R} \cup \mathcal{H}$ is terminating. If $\mathrm{Expd}(\mathcal{R}, \mathcal{H}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{H}}$, then $\mathcal{R} \vdash_i \mathcal{H}$.* ∎

Theorem 5.2.11. *If $\mathcal{S}^n_{RI}[\![\mathcal{R}]\!] = \varnothing$ for some $\mathcal{R} \in \mathfrak{R}(\mathcal{C})$, then $\mathcal{R}_0 \vdash_i \mathcal{E}$.*

Proof. Similar to Theorem 5.2.8, using Lemma 5.2.10. ∎

Example 5.2.12. *Consider the following TRS $\mathcal{R}_0$:*

$$1: \quad x + 0 \to 0 \qquad\qquad 2: \quad x + \mathsf{s}(y) \to \mathsf{s}(x + y)$$

and $\mathcal{E}$ the single equality

$$3: \quad (x + y) + z \approx x + (y + z)$$

As previously, we assume the signature to be sorted, here with $0 : \mathrm{nat}, \mathsf{s} : \mathrm{nat} \to \mathrm{nat}$ and $+ : \mathrm{nat} \times \mathrm{nat} \to \mathrm{nat}$. We obtain the initial constraints $\mathcal{C}_0 = \{(3, \top)\}$. Suppose $R(\mathcal{C}_0)$ is the singleton $\mathcal{R}_0 = \{1, 2, 3\}$. Then expanding 3 results in

$$\mathcal{C}_1 = S_{RI}(\mathcal{C}_0) = \{(3, \neg\mathcal{R}_0), (4, \top)\}$$

where 4 is

$$4: \quad x + z \approx x + (0 + z) \qquad\qquad 5: \quad \mathsf{s}(x + y) + z = x + (\mathsf{s}(y) + z)$$

In fact it is not difficult to see that orienting 5 and 3 will lead to an infinite run. However, suppose now that $R(\mathcal{C}_1)$ is the singleton $\mathcal{R}_1 = \{1, 2, 3'\}$. Expansion leads to

$$\mathcal{C}_1 = S_{RI}(\mathcal{C}_0) = \{(3, \neg\mathcal{R}_0 \vee \neg\mathcal{R}_1), (4, \neg\mathcal{R}_0), (5, \mathcal{R}_0)\}$$

With $\mathcal{R}_1$, we have $S_{RI}(\mathcal{C})[\![\mathcal{R}_1]\!] = \varnothing$. One can verify that $\mathcal{R}_0$ is quasi-reducible, and the position 1 of the rhs of equation 3 is basic. Thus $\mathcal{R}_0 \vdash_i \mathcal{E}$.

Example 5.2.12 illustrates two important facts: First, similar as in Knuth-Bendix completion, the orientation of an equation to a rule is crucial, and a bad choice can easily lead to divergence. Secondly, in practice it is not necessary to exhaustively compute all TRSs at once. A bad choice can be always recovered at later stage.

5.3 Automation

The challenge when automating our framework is the question how to find a set of terminating TRSs $\mathfrak{R}(\mathcal{C})$ given a set of constrained equalities $\mathcal{C}$.

As introduced in Chapter 4, SAT/SMT-encodings of termination conditions based on existing subclasses of reduction orders are well established nowadays, cf. [52, 17, 87, 23]. All of them can test the existence of a tuple of an assignment of boolean variables and a reduction order $(M, \succ)$ that satisfies arbitrary Boolean combinations of boolean variables and *order constraints*:

$$C ::= s > t \mid x \mid \top \mid \bot \mid \neg C \mid C \vee C \mid C \wedge C$$

For the base cases, we define $(M, \succ) \models \ell > r$ if $\ell \succ r$ and $(M, \succ) \models x$ if x is true in M, and extended this inductively to arbitrary formulas. Note the difference to the formulation in Chapter 4 where the second base case does not exist and the constraints allow no additional boolean variables.

All encodings of termination conditions allow to further automatically translate constraints of form $s > t$ with respect to a class of reduction orders to Boolean (or SMT) constraints. Therefore one can treat order constraints as an instance of MaxSAT/MaxSMT. When encoding order constraints w.r.t. specific classes of reduction orders, one needs to carefully distinguish termination constraints and order constraints. We now show how to translate termination constraints to order constraints.

Let ψ be a termination constraint. We write $\mathsf{Rules}(\psi)$ for the set of all rules that appear in ψ. We associate with each rule $\ell \to r \in \mathsf{Rules}(\psi)$ a fresh boolean variable $x_{\ell\to r}$, and write $\widehat{\psi}$ for the constraint generated by replacing each $\ell \to r \in \mathsf{Rules}(\psi)$ by the associated variable $x_{\ell\to r}$ in ψ. Then with the next lemma, we can translate satisfiability of a termination constraint problem to satisfiability of an order constraint problem.

Lemma 5.3.1. *Let ψ be a termination constraint, and $\mathcal{R}$ and $\mathcal{R}'$ terminating TRSs with $\mathcal{R} \subseteq \mathcal{R}'$ and $\mathsf{Rules}(\psi) \subseteq \mathcal{R}'$. Then*

$$\mathcal{R} \models \psi \quad \textit{if and only if} \quad (M, \succ) \models \widehat{\psi} \wedge \bigwedge_{\ell\to r\in\mathcal{R}'} (\neg x_{\ell\to r} \vee \ell > r) \wedge \bigwedge_{\ell\to r\in\mathcal{R}} x_{\ell\to r}$$

for some model $(M, \succ)$.

Proof. For the "only-if" direction, we can construct a model $(M, \succ)$ from $\mathcal{R}$ as follows. Let $\succ$ denote $\to^+_{\mathcal{R}}$. Define $(M, \succ) \models x_{\ell\to r}$ if $\ell \to r \in \mathcal{R}$, and $(M, \succ) \not\models x_{\ell\to r}$ otherwise. Moreover define $(M, \succ) \models \ell \to r$ if $\ell \succ r$. The "if" direction is very analogous of the proof of Lemma 5.3.2. ∎

With Lemma 5.3.1, we can automate the search for a suitable TRS, but for the encoding we still need to fix the domain of the — to be computed — TRS $\mathcal{R}$.

We usually take as the domain all possible orientations of the equational part of $\mathcal{C}$ together with all rules appearing in their associated constraints. In Rewriting Induction, often additional lemmata are needed for a successful proof. If *sound* lemmata have been found by other means, one can extend the domain of $\mathcal{R}$ accordingly.

In general there is no complete way to find a suitable TRS among candidates. However given a set of constrained equations $\mathcal{C}$, the goal is to find a TRS $\mathcal{R}$, such that $\mathcal{C}[\![\mathcal{R}]\!] = \varnothing$. Thus optimally $\mathcal{R} \models \neg C$ for all $(s \approx t, C) \in \mathcal{C}$. If $\mathcal{R} \models C$ for some $(s \approx t, C) \in \mathcal{C}$, it makes sense to try to orient $s \approx t$, since then the equation is at least joinable in the next step. We try to solve the joinability problem of a CES $\mathcal{C}$ by iteratively enumerating candidate TRSs. Similarly to Chapter 4 we write

$$\text{Maximize} \quad S \quad \text{subject to } \varphi$$

for the problem of finding an assignment that maximizes the number of satisfied elements in the set S while satisfying φ. Given k (failed, i.e. $\mathcal{C}[\![\mathcal{R}]\!] \neq \varnothing$) TRSs $\mathcal{R}_1, \ldots, \mathcal{R}_k$, we solve the following maximal satisfaction problem ψ over termination constraints:

$$\textit{Maximize} \quad \{\neg C \vee s \to t \vee t \to s\}_{(s\approx t, C)\in\mathcal{C}} \quad \textit{subject to} \quad \bigwedge_{i=1}^{k} \neg \bigwedge \mathcal{R}_i$$

This can be translated to the following satisfaction problem χ over order constraints:

$$\begin{aligned} &\textit{Maximize} \quad \{\neg \widehat{C} \vee x_{s\to t} \vee x_{t\to s}\}_{(s\approx t, C)\in\mathcal{C}} \quad \textit{subject to} \\ (1) \qquad &\bigwedge_{i=1}^{k} \neg \Big(\bigwedge_{\ell\approx r\in\mathcal{R}_i} x_{\ell\to r} \Big) \wedge \bigwedge_{\ell\to r\in \mathsf{Rules}(\psi)} \neg x_{\ell\to r} \vee \ell > r \end{aligned}$$

For automation we need to construct a terminating TRS that is a maximal solution of the termination constraint ψ out of a model that maximizes the above order constraint χ. The next lemma illustrates how this can be done.

Lemma 5.3.2. *Suppose* $(M, \succ) \models \chi$. *Then*

$$\{\ell \to r \in \mathsf{Rules}(\psi) \mid (M, \succ) \models x_{\ell\to r}\} \models \psi.$$

Proof. Let $\mathcal{R} = \{\ell \to r \in \mathsf{Rules}(\psi) \mid (M, \succ) \models x_{\ell\to r}\}$. Due to the part

$$\bigwedge_{\ell\to r\in\mathsf{Rules}(\psi)} \neg x_{\ell\to r} \vee \ell > r$$

of constraint (1), satisfaction of $x_{\ell\to r}$ implies $\ell \succ r$ for all $\ell \to r \in \mathcal{R}$. Due to $\bigwedge_{i=1}^{k} \neg(\bigwedge_{\ell\approx r\in\mathcal{R}_i} x_{\ell\to r})$ of (1), we have $\mathcal{R} \neq \mathcal{R}_i$ for all $1 \leq i \leq k$. One can verify that $\mathcal{R}$ is indeed also a maximal solution to ψ. ∎

If the satisfaction problem is not satisfiable, we have exhaustively computed all potential TRSs over the candidate equations w.r.t. the considered class of reduction orders.

5.4 Experiments

We implemented both completion and rewriting induction. In both cases, the SMT solver Yices [22] is used to solve ordered constraint problems. Encoded is the class of all lexicographic path orders and the class of all Knuth-Bendix path orders. The tests were single-threaded run on a system equipped with an Intel Core Duo L7500 with 1.6 GHz and 2 GB of RAM, using a timeout of 10 seconds for rewriting induction, and a timeout of 60 seconds for completion. A full list of all results can be found in Appendix B.

Completion

A straight-forward implementation of completion fails. The set of critical pairs is exponential at worst, and maximizing the set of oriented equations yields exponential growth in practice. For example for group-theory [49], which consists of only three equations initially, after a few iterations several hundred equations are contained in $\mathcal{C}$.

Moreover, some equations tend to generate a lot of critical pairs, a problem which all completion procedures face. Several selection heuristics have been proposed to avoid such equations. In our implementation we use a similar approach. Instead of taking the whole equational part of $\mathcal{C}$ as candidates for the TRS $\mathcal{R}$, we only select certain equations from $\mathcal{C}$. We write $\mathcal{C}^{<d}$ for

$$\{(s \approx t, C) \in \mathcal{C} \mid |s| + |t| < d\}$$

and we write $\mathcal{C}\restriction_n$ for the set of the n smallest (w.r.t. the size of the equational part) constrained equalities in $\mathcal{C}$. We refine the encoding from Section 4.2 to the following maximality problem over termination constrains:

$$\textit{Maximize} \quad \{\neg C \vee s \to t \vee t \to s\}_{(s\approx t,C)\in(\mathcal{C}^{<d})\restriction_n} \quad \textit{subject to} \quad \bigwedge_{i=1}^{k} \neg \bigwedge \mathcal{R}_i$$

In our implementation we fix $d = 20$, we start with parameter n set to 7, and increase n by 7 with each iteration. These values were experimentally chosen; larger parameters could not be processed by our tool.

One should remark at this point the difference between Maximal Completion (Chapter 4), and the current approach. In Maximal Completion, equations in $\mathcal{S}(\mathcal{C})$ are directly reduced. The condition for soundness of the procedure — specific to completion — needs to be ensured separately. Here however, the conditions for soundness are directly encoded in the constraint part of $\mathcal{S}(\mathcal{C})$.

The test-bed consists of 115 equational systems contained in the distribution of mkbTT.[1] Table 5.1 gives a summary of the overall results. For comparison we also include results from Maxcomp, for comparison here with a timeout of 60 seconds.

[1] http://cl-informatik.uibk.ac.at/software/mkbtt/

Table 5.1: Summary for 115 equational systems

	LPO		KBO	
	Maxcomp	Constraints	Maxcomp	Constraints
completed	84	47	69	33
failure	6	6	3	3
timeout	25	62	33	79

The experiments clearly show that maximal completion is more effective than the current constraint based approach. From our observation there are two reasons for it. The first one is that maintaining termination constraints for each equation and evaluating them creates some non-trivial overhead. The second reason however is *maximality*. In the approach based on constrained equations we optimize the selection of a TRS among candidate equations with respect to the number of unjoined equations. Thus equations that would create additional critical pairs will not be selected. Theoretically this should increase the performance.

However maximality ensures that adding such equations can never be harmful. And in practice in a lot of cases it is quite beneficial to add such equations because it decreases the chance to miss an important equation, i.e. the selection strategy becomes less important as illustrated in Example 4.3.1. Moreover the additional time spent for constructing additional critical pairs is marginal.

Rewriting Induction

The set Expd is a subset of all critical pairs. Thus, in Rewriting Induction, the set $\mathcal{C}$ grows much slower compared to Completion, and no specific selection is necessary. In fact, in practice it rarely happens that the proof fails due to computational resources. Rather, a failed proof attempt is due to other reasons, such as the need for additional lemmata. Thus our implementation follows the approach described in Section 4.2.

The test-bed consists of 73 (valid) inductive conjectures from the (no longer maintained) *Dream Corpus of Inductive Conjectures*, and from [66]. Results of a prototype implementation are depicted in Table 5.2.

To our best knowledge there are no fully automated inductive theorem provers except for [66], which is an adaptation of multi-completion to inductive theorem

Table 5.2: Summary for 73 inductive conjectures

	LPO	KBO
success	32	21
timeout	41	52

proving, called multi-context rewriting induction. They report 35 solved conjectures on the Dream Corpus. The combined number of examples solved by our approach using the class of LPOs and KBOs on the Dream Corpus is 32. Thus our approach has a quite comparable performance. Moreover multi-context rewriting induction uses a termination tool to construct a reduction order on the fly. Of course, this is more powerful than a constraint encoding of all LPOs and KBOs. While no detailed data are provided in [66] we highly suspect that the small difference in proven conjectures can be mostly attributed to that. One has to clearly state that despite its benefits the fact that not all methods of showing termination can be encoded to constraints is the biggest drawback with our method.

5.5 Related Work

We compare existing frameworks and their proof techniques with our framework based on constrained equalities. For existing frameworks see also Chapter 3.

- **Completion**. Proving soundness of completion procedures is intricate, especially when inter-reduction is used. The first complete proof of Knuth and Bendix' procedure including inter-reduction was due to Huet [40] and appeared roughly eleven years afterwards. Moreover it only shows soundness for one particular implementation of the algorithm, and even minor modifications can easily violate soundness [69, 75]. In order to address this problem, Dershowitz, Plaisted, Bachmair and Hsiang [12, 11] recast Huet's completion in inference rules. They employed a new proof technique called *proof orderings*, which significantly simplifies soundness-proofs in completion. Due to its universality, nowadays this framework is the most popular way to formalize various completion procedures, including ordered completion, inductionless induction, and rewriting induction. In our soundness proof, simple reductionism of joinability problems is used instead of the proof ordering method.

- **Multi-Completion** [51]. The above framework of inference rules requires a fixed reduction order, which is used to show the necessary termination of the generated systems. Since it is hard to know a suitable order a priori, often one has to restart the process with another order, which usually requires user interactions. Kondo and Kurihara introduced an efficient data structure, the *node*, to run completion procedures with different orders in parallel. Adaptations are highly efficient and have been used for several settings ([66, 83]). However each adaptation requires careful formulation as well as its soundness proof which seems a non-trivial task. Usually, soundness is established by extracting a single run out of the procedure, and then applying the proof-ordering approach.

- **Maximal Completion** [46]. The original maximal completion can be simulated if $\top$ is always associated to equalities. Inter-reduction was not available in maximal completion but by using constrained equalities we can fully recover this functionality. Folklore suggests that inter-reduction improves efficiency in completion, but in our approach this is not the case as seen in experimental data. This is because rather than reducing terms, avoiding a diverging processes is more critical for a successful run. Moreover, by not using inter-reduction, more equalities are shared among different orderings, which avoids missing suitable equalities. The overhead costs, i.e. the increased amount of time needed for processing equalities are neglectable compared to other costs.

  Soundness in Maximal Completion is ensured directly from Knuth and Bendix' criterion (Theorem 2.4.5). This, together with a lack of explicit configurations — which are needed to maintain the joinability requirements in inductive reasoning — prohibit to formulate other procedures in it.

We conclude this chapter by emphasizing the advantages of our framework based on constrained equalities. For one, it is universal enough to formulate several completion procedures with minimal effort. Here we showed formulations for completion, inductionless induction and rewriting induction. But other procedures can be easily expressed as well by varying the parameters $\mathcal{S}$ and F. Secondly, proofs for soundness can easily be established using reductionism. Lastly, experiments indicate practical feasibility. One has to note however, that Maximal Completion's practical strength is unparalleled, and future work should investigate data-structures to improve scalability, as done in multi-completion.

Chapter 6

Confluence and Relative Termination

For terminating TRSs, confluence follows from local confluence, which itself can be tested by the joinability of critical pairs. Since the construction of critical pairs relies purely on syntactical properties, critical pairs can be effectively computed, and a confluence test can be fully automated. For non-terminating TRSs, confluence is undecidable. For non-terminating but left-linear TRSs, confluence criteria can be given by restricting the shape of the join-relation. Essential here is that left-linearity allows to trace untouched redexes over several rewrite steps using the notion of *residuals* (cf. Chapter 4 of [75]). Together with the absence of (non-trivial) critical pairs, this allows to derive effectively automatable confluence criteria.

Unfortunately, for non-terminating and non-left-linear TRSs, none of these approaches work. The following example, due to Huet, gives an intuition for the underlying problems:

Example 6.0.1 ([39])**.** *Consider the following TRS:*

$$1\colon\ \mathsf{f}(x,x) \to \mathsf{a} \qquad 2\colon\ \mathsf{f}(x,\mathsf{g}(x)) \to \mathsf{b} \qquad 3\colon\ \mathsf{c} \to \mathsf{g}(\mathsf{c})$$

which is non-confluent since

$$\mathsf{a} \leftarrow \mathsf{f}(\mathsf{c},\mathsf{c}) \to \mathsf{f}(\mathsf{c},\mathsf{g}(\mathsf{c})) \to \mathsf{b}$$

and a *and* b *are not joinable.*

Joinability of critical pairs does not imply confluence in the absence of termination, as can be inferred from Example 6.0.1, where the set of critical pairs is empty. In orthogonal systems, rewriting residuals of one step after another step or vice-versa results in the same end term, but non-left-linear rules destroy this property. Thus, in the case of non-left-linear and non-terminating TRSs, not only can none of the abstract notions like local confluence be utilized, also none of the technical tools, such as (syntactical) critical pairs or residuals are applicable, and no established tool-set is available.

Therefore one has to fundamentally modify these notions to yield new criteria. Existing approaches [31, 44] use an extended notion of critical pairs to formulate suitable joinability conditions. These extensions however come with a price to pay, as they require equational unification, which itself is undecidable. Thus even if sound criteria can be formulated with these extended notions, extra care is needed to ensure computability. For example, Jouannaud and Kirchner's result is practically limited to TRSs where associativity and commutativity (AC) are the cause of non-termination, since there, effective algorithms to solve AC-unification exist. Gomi et al. developed a sound, but incomplete approximation algorithm to practically apply their criterion.

In Section 6.1, we present a new confluence criterion for non-left-linear and non-terminating TRSs, which is based on (modest) syntactic restrictions and relative termination, and show its application with some examples. Then we give a proof in Section 6.2. Our criterion also requires joinability of extended critical pairs. However we provide a result on unification in Section 6.3, that ensures that joinability can be tested by computing syntactic unifiers only. With it we develop a fully automatic procedure and evaluate it experimentally in Section 6.4. Finally our result is compared with existing works in Section 6.5.

6.1 Confluence Criterion

We first need a few non-standard notions in term rewriting to state our main result. We write $\to_1/\to_2$ for $\to_2^* \cdot \to_1 \cdot \to_2^*$, and $\to_{\mathcal{R}/\mathcal{S}}$ for $\to_{\mathcal{R}}/\to_{\mathcal{S}}$.

Definition 6.1.1. *A TRS $\mathcal{R}$ is* relatively terminating *over a TRS $\mathcal{S}$ (we also say $\mathcal{R}/\mathcal{S}$ is* terminating*), if $\to_{\mathcal{R}/\mathcal{S}}$ is terminating.*

In Definition 2.4.4, we defined syntactical critical pairs. We now give an extended notion of overlaps and critical pairs. An *extended rewrite rule* is a pair (ℓ, r) of terms with $\ell \notin \mathcal{V}$, and an *extended TRS* (*eTRS*) is a set of extended rewrite rules.

Definition 6.1.2. *Let $\mathcal{R}_1, \mathcal{R}_2, \mathcal{S}$ be eTRSs. An $\mathcal{S}$-overlap $(\ell_1 \to r_1, p, \ell_2 \to r_2)_\sigma$ of $\mathcal{R}_1$ on $\mathcal{R}_2$ consists of a variant $\ell_1 \to r_1$ of a rule in $\mathcal{R}_1$ and a variant $\ell_2 \to r_2$ of a rule in*

$\mathcal{R}_2$, a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, and a substitution σ, such that the following conditions hold:

- *$\ell_1\sigma \leftrightarrow_{\mathcal{S}}^* \ell_2|_p\sigma$, and*
- *if $p = \varepsilon$, then $\ell_1 \to r_1$ and $\ell_2 \to r_2$ may not be variants of each other*

The pair $(\ell_2\sigma[r_1\sigma]_p, r_2\sigma)$ induced from the overlap is an $\mathcal{S}$-extended critical pair *(or simply* $\mathcal{S}$-critical pair*) of $\ell_1 \to r_1$ and $\ell_2 \to r_2$ at p, written*

$$\ell_2\sigma[r_1\sigma]_p \; {}_{\mathcal{R}_1}\!\leftarrow_{\mathcal{S}}{}^{\infty}\!\rightarrow_{\mathcal{R}_2} r_2\sigma.$$

We will simply write $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})$ for ${}_{\mathcal{R}}\!\leftarrow_{\mathcal{S}}{}^{\infty}\!\rightarrow_{\mathcal{R}}$, namely the $\mathcal{S}$-critical pairs originating from $\mathcal{S}$-overlapping rules from $\mathcal{R}$ on $\mathcal{R}$.

Example 6.1.3 (continued from Example 6.0.1)**.** *Consider again Example 6.0.1 and the TRSs $\mathcal{R} = \{1, 2\}$ and $\mathcal{S} = \{3\}$. Then e.g.*

$$(\, \mathsf{f}(x, x) \to \mathsf{a},\ \epsilon,\ \mathsf{f}(x, \mathsf{g}(x)) \to \mathsf{b}\,)_{\{x \mapsto \mathsf{c}\}}$$

is an $\mathcal{S}$-overlap of $\mathcal{R}$. Its induced critical pair is (b, a).

It should be remarked that the definition of ($\mathcal{S}$-)critical pairs includes pairs originating from non-minimal unifiers. Usually, they are excluded to guarantee finiteness of the set of critical pairs. Thus, for example taking the previous Definition 2.4.4, even when considering only a TRS $\mathcal{R}$ (and no eTRS), the inclusion $\mathsf{CP}_{\varnothing}(\mathcal{R}) \subseteq \mathsf{CP}(\mathcal{R})$ does not hold in general. However due to the existence of mgu's for critical pairs (see Section 6.3), we may restate Knuth and Bendix' criterion (Theorem 2.4.5) as: A terminating TRS is confluent if and only if $\mathsf{CP}_{\varnothing}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}}$.

Let $\mathsf{REN}(t)$ denote a linear term resulting from replacing in t each variable occurrence by a fresh variable. We write $\widehat{\mathcal{R}}$ for the eTRS

$$\{\mathsf{REN}(\ell) \to r \mid \ell \to r \in \mathcal{R}\}.$$

Definition 6.1.4. *A TRS $\mathcal{S}$ is* strongly non-overlapping *on $\mathcal{R}$ if $\widehat{\mathcal{S}}$ has no (syntactic) overlaps on $\widehat{\mathcal{R}}$. We write $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$ if both $\mathcal{S}$ is strongly non-overlapping on $\mathcal{R}$, and $\mathcal{R}$ is strongly non-overlapping on $\mathcal{S}$.*

Example 6.1.5 (continued from Example 6.0.1)**.** *Consider again Example 6.0.1. We have $\mathsf{SNO}(\{1, 2\}, \{3\})$. But $\mathsf{SNO}(\{1\}, \{2\})$ does not hold, since*

$$\widehat{\{1\}} = \{\mathsf{f}(x_1, x_2) \to \mathsf{a}\} \text{ and } \widehat{\{2\}} = \{\mathsf{f}(x_3, \mathsf{g}(x_4)) \to \mathsf{b}\}$$

are syntactically overlapping, as $\mathsf{f}(x_1, x_2)$ and $\mathsf{f}(x_3, \mathsf{g}(x_4))$ unify.

We have now all the definitions to state our main theorem.

Theorem 6.1.6. *Suppose that $\mathcal{S}$ is confluent, $\mathcal{R}/\mathcal{S}$ is terminating, and $\mathsf{SNO}(\mathcal{R},\mathcal{S})$. The union $\mathcal{R}\cup\mathcal{S}$ of the TRSs is confluent if and only if $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})\subseteq{\downarrow_{\mathcal{R}\cup\mathcal{S}}}$.* ∎

Note that the theorem is a proper generalization of Knuth and Bendix' confluence criterion when $\mathcal{S}\neq\varnothing$, and coincides with it when $\mathcal{S}=\varnothing$. The next examples illustrate Theorem 6.1.6. Note that no existing tool can prove confluence of the involved TRSs automatically (see Section 6.4).

Example 6.1.7. *Consider the TRS*

$$1\colon\ \mathsf{f}(x,x)\to(x+x)+x \qquad\qquad 2\colon\ x+y\to y+x$$

Take $\mathcal{R}=\{1\}$ and $\mathcal{S}=\{2\}$. One can easily verify $\mathsf{SNO}(\mathcal{R},\mathcal{S})$. Termination of $\mathcal{R}/\mathcal{S}$ can be established using a termination tool such as T$_{\text{T}}$T2 v1.06 [50][1]*, and confluence of $\mathcal{S}$ follows from orthogonality. Because of $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})=\varnothing\subseteq{\downarrow_{\mathcal{R}\cup\mathcal{S}}}$, we conclude confluence by Theorem 6.1.6.*

Example 6.1.8. *Consider the TRS*

$$1\colon\ \mathsf{f}(x,x)\to\mathsf{s}(\mathsf{s}(x)) \qquad\qquad 2\colon\ \infty\to\mathsf{s}(\infty)$$

Take $\mathcal{R}=\{1\}$ and $\mathcal{S}=\{2\}$. As in Example 6.1.7, one can easily verify the conditions of Theorem 6.1.6, including $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})=\varnothing\subseteq{\downarrow_{\mathcal{R}\cup\mathcal{S}}}$. Hence the TRS is confluent.

Example 6.1.9. *Consider the TRS*

$$\begin{array}{lll} 1\colon\ \mathsf{eq}(\mathsf{s}(n),x:xs,x:ys)\to\mathsf{eq}(n,xs,ys) & & 3\colon\ \mathsf{nats}\to 0:\mathsf{inc}(\mathsf{nats}) \\ 2\colon\ \mathsf{eq}(n,xs,xs)\to\mathsf{T} & & 4\colon\ \mathsf{inc}(x:xs)\to\mathsf{s}(x):\mathsf{inc}(xs) \end{array}$$

Take $\mathcal{R}=\{1,2\}$ and $\mathcal{S}=\{3,4\}$. Again, $\mathsf{SNO}(\mathcal{R},\mathcal{S})$, termination of $\mathcal{R}/\mathcal{S}$ and confluence of $\mathcal{S}$ is established. Moreover, one can show

$$\mathsf{CP}_{\mathcal{S}}(\mathcal{R})\ =\ \{(\mathsf{eq}(s,t,u),\mathsf{T})\mid s,\,t,\,u\text{ are terms and }t\leftrightarrow_{\mathcal{S}}^{*}u\}$$

and thus the set is included in $\downarrow_{\mathcal{R}\cup\mathcal{S}}$ because of confluence of $\mathcal{S}$. Hence by using Theorem 6.1.6 we infer that $\mathcal{R}\cup\mathcal{S}$ is confluent.

We conclude this section by mentioning that all conditions of Theorem 6.1.6 are essential. One cannot drop $\mathsf{SNO}(\mathcal{R},\mathcal{S})$ nor replace joinability of $\mathcal{S}$-critical pairs by joinability of critical pairs.

[1] `http://colo6-c703.uibk.ac.at/ttt2/`

Example 6.1.10 (continued from Example 6.0.1)**.** *Recall that the TRS in Huet's example consists of*

$$1\colon\ \mathsf{f}(x,x) \to \mathsf{a} \qquad 2\colon\ \mathsf{f}(x,\mathsf{g}(x)) \to \mathsf{b} \qquad 3\colon\ \mathsf{c} \to \mathsf{g}(\mathsf{c})$$

which is non-confluent. If one takes $\mathcal{R} = \{1\}$ *and* $\mathcal{S} = \{2,3\}$*, then* $\mathcal{R}/\mathcal{S}$ *is terminating,* $\mathcal{S}$ *is confluent, and* $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) = \varnothing \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$*, however* $\mathsf{SNO}(\mathcal{R},\mathcal{S})$ *is violated. If one takes* $\mathcal{R} = \{1,2\}$ *and* $\mathcal{S} = \{3\}$ *then,* $\mathsf{SNO}(\mathcal{R},\mathcal{S})$*,* $\mathcal{R}/\mathcal{S}$ *is terminating,* $\mathcal{S}$ *is confluent and* ${}_{\mathcal{R}}{\leftarrow}{\infty}{\rightarrow}_{\mathcal{R}} = \varnothing \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$*. Yet the non-joinable* $\mathcal{S}$*-critical pair* $\mathsf{a}\ {}_{\mathcal{R}}{\leftarrow}{\infty}{\rightarrow}_{\mathcal{R}}\ \mathsf{b}$ *exists.*

Also, termination of $\mathcal{R}/\mathcal{S}$ is essential. We illustrate it with another famous non-confluent and non-left-linear TRS, due to Klop:

Example 6.1.11. *Consider Klop's example [48]*

$$1\colon\ \mathsf{f}(x,x) \to \mathsf{a} \qquad 2\colon\ \mathsf{g}(x) \to \mathsf{f}(x,\mathsf{g}(x)) \qquad 3\colon\ \mathsf{c} \to \mathsf{g}(\mathsf{c})$$

which is known to be non-confluent, since

$$\mathsf{a} \leftarrow \mathsf{f}(\mathsf{g}(c),\mathsf{g}(c)) \leftarrow \mathsf{f}(c,\mathsf{g}(c)) \leftarrow \mathsf{g}(\mathsf{c}) \to \mathsf{g}(\mathsf{g}(\mathsf{c})) \to^* \mathsf{g}(\mathsf{a})$$

and a *and* $\mathsf{g}(\mathsf{a})$ *are not joinable. If one would take* $\mathcal{R} = \{1,2\}$ *and* $\mathcal{S} = \{3\}$*, then* $\mathsf{SNO}(\mathcal{R},\mathcal{S})$*,* $\mathcal{S}$ *is confluent, and* $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) = \varnothing \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$*. However* $\mathcal{R}/\mathcal{S}$ *is non-terminating, since rule number* 2 *is looping.*

6.2 Correctness of the Criterion

We now give a proof of the main theorem 6.1.6. In the proof we will make use of the *decreasing diagram technique* [79, 81] of van Oostrom, a powerful confluence criterion for abstract rewrite systems.

Let $\mathcal{A} = (A, \langle \to_\alpha \rangle_{\alpha\in I})$ be an ARS and $>$ a proper order on I. For every $\alpha \in I$ we write $\overset{\vee}{\to}_\alpha$ for

$$\{\to_\beta |\ \beta \in I \text{ and } \beta < \alpha\},$$

and $\overset{\vee}{\to}{}^*_\alpha$ for $(\overset{\vee}{\to}_\alpha)^*$. The union of $\overset{\vee}{\to}_\alpha$ and ${}_\alpha\overset{\vee}{\leftarrow}$ is denoted by $\overset{\vee}{\leftrightarrow}_\alpha$. For $\alpha,\beta \in I$, the union of $\overset{\vee}{\to}_\alpha$ and $\overset{\vee}{\to}_\beta$ is written as $\overset{\vee}{\to}_{\alpha\beta}$.

Definition 6.2.1. *Two labels* α *and* β *are* decreasing *with respect to* $>$ *if*

$${}_\alpha{\leftarrow}\cdot{\rightarrow_\beta} \subseteq {\overset{\vee}{\leftrightarrow}}^*_\alpha \cdot {\to}^{=}_\beta \cdot {\overset{\vee}{\leftrightarrow}}^*_{\alpha\beta} \cdot {}^{=}_\alpha{\leftarrow} \cdot {\overset{\vee}{\leftrightarrow}}^*_\beta$$

Decreasingness is illustrated in Figure 6.1. An ARS $\mathcal{A} = (A, \langle \to_\alpha \rangle_{\alpha\in I})$ *is* decreasing *if there exists a well founded order* $>$ *such that every two labels in* I *are decreasing with respect to* $>$*.*

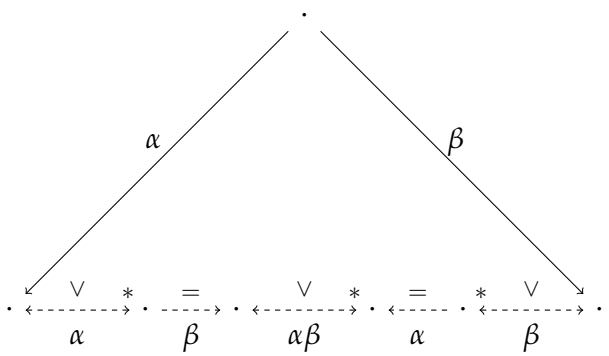

Figure 6.1: Decreasingness

Decreasingness allows to formulate a very powerful confluence criterion on ARSs.

Theorem 6.2.2 ([81]). *Every decreasing ARS is confluent.* ∎

We recall an example from [81] to illustrate how to use the decreasing diagram technique to show confluence of TRSs.

Example 6.2.3. *Recall the "if"-direction of Lemma 2.1.6 (Newman), which, when applied to TRSs, states that a terminating and locally confluent TRS is confluent. Let $\mathcal{R}$ be a terminating and locally confluent TRS. To apply Theorem 6.2.2, we need to associate an ARS with the TRS $\mathcal{R}$. We will consider the ARS $\mathcal{A}$ with the set A as $\mathcal{T}(\mathcal{F},\mathcal{V})$, and we introduce labels by* labelling *rewrite steps by its* source*: We label a step $s \to t$ as $s \to_s t$. Lastly we need to compare labels by an ordering $>$, here we choose $> := \to_{\mathcal{R}}^{+}$. Since $\mathcal{R}$ is terminating, $>$ is a well-founded order.*

Now consider any peak ${}_s\!\leftarrow \cdot \to_v$. By the definition of source labelling we have $v = s$, i.e. ${}_s\!\leftarrow \cdot \to_s$, and there exist u and t such that $u \leftarrow s \to t$. By local confluence of $\mathcal{R}$, there exists a sequence

$$u \to_{\mathcal{R}} u_1 \to_{\mathcal{R}} \ldots \to_{\mathcal{R}} u_n \;{}_{\mathcal{R}}\!\leftarrow t_m \;{}_{\mathcal{R}}\!\leftarrow \ldots {}_{\mathcal{R}}\!\leftarrow t_1 \;{}_{\mathcal{R}}\!\leftarrow t$$

and thus a labelled sequence

$$u \to_u u_1 \to_{\mathcal{R}} \ldots \to_{u_{n-1}} u_n \;{}_{t_m}\!\leftarrow t_m \;{}_{t_{m-1}}\!\leftarrow \ldots {}_{t_1}\!\leftarrow t_1 \;{}_{t}\!\leftarrow t.$$

We have $s > u$ and $s > u_i$ for all $1 \leq i < n$, and $s > t$ and $s > t_j$ for all $1 \leq j \leq m$. Therefore $\mathcal{A}$ is decreasing, and thus by Theorem 6.2.2 confluent. Moreover confluence of $\mathcal{A}$ readily implies confluence of $\mathcal{R}$.

In our proof, we do not directly show confluence of the rewrite relation. Let $\mathcal{R}$ and $\mathcal{S}$ be TRSs. We introduce an *intermediate* relation $\twoheadrightarrow$, such that $\to_{\mathcal{R}\cup\mathcal{S}} \subseteq \twoheadrightarrow \subseteq \to_{\mathcal{R}\cup\mathcal{S}}^{*}$. Confluence of this intermediate relation readily implies confluence

of $\mathcal{R} \cup \mathcal{S}$. The relation $\twoheadrightarrow$ is defined as the union of $\to_{\mathcal{R}_\mathcal{S}}$ and $\to^*_\mathcal{S}$, where $\mathcal{R}_\mathcal{S}$ is the TRS

$$\{\ell'\sigma \to r\tau \mid \ell'\rho \to r \in \mathcal{R} \text{ and } \sigma \to^*_\mathcal{S} \rho\tau \text{ for some variable substitution } \rho\ \}$$

Here $\sigma \to^*_\mathcal{S} \tau$ means that $x\sigma \to^*_\mathcal{S} x\tau$ for all variables x.

Remark 6.2.4. *It is important to note that*

a) in the definition of $\mathcal{R}_\mathcal{S}$, linearity of ℓ' can be assumed without loss of generality, and

b) the inclusions $\to_\mathcal{R} \subseteq \to_{\mathcal{R}_\mathcal{S}} \subseteq \to^*_\mathcal{S} \cdot \to_\mathcal{R}$ hold.

We show confluence of $\twoheadrightarrow$ by the decreasing diagram technique with the predecessor labeling [81]: We write $b \twoheadrightarrow_a c$ if $a \twoheadrightarrow^* b \twoheadrightarrow c$. Labels are compared with respect to $\to^+_{\mathcal{R}/\mathcal{S}}$, denoted by $>$. Since termination of $\mathcal{R}/\mathcal{S}$ is presupposed in the theorem, the relation $>$ forms a well-founded order.

The next lemma states a general property of rewriting in substitutions.

Lemma 6.2.5. *Let $\mathcal{R}$ be a TRS. If $t\sigma \xrightarrow{p}_\mathcal{R} u$ and $p \notin \mathcal{P}\mathsf{os}_\mathcal{F}(t)$ then $u \to^*_\mathcal{R} t\tau$ for some τ with $\sigma \to^=_\mathcal{R} \tau$.*

Proof. Suppose $t\sigma \xrightarrow{p}_\mathcal{R} u$ and $p \notin \mathcal{P}\mathsf{os}_\mathcal{F}(t)$. Then there exists a variable position $q \in \mathcal{P}\mathsf{os}_\mathcal{V}(t)$ with $q \leqslant p$ and $u = (t\sigma)[u|_q]_q$. Let Q be the set of all positions of $t|_q$ in t. Since $u|_{q'} \to_\mathcal{R} u|_q$ holds for all $q' \in Q \setminus \{q\}$, we have $u \to_\mathcal{R} (t\sigma)[u|_q]_{q' \in Q \setminus \{q\}}$. The latter term is identical to $(t\sigma)[u|_q]_{q' \in Q}$. We define the substitution τ as follows:

$$\tau(x) = \begin{cases} u|_q & \text{if } x = t|_q \\ x\sigma & \text{otherwise} \end{cases}$$

One can verify $\sigma \to^=_\mathcal{R} \tau$ and $(t\sigma)[u|_q]_{q' \in Q} = t\tau$. Hence $u \to^*_\mathcal{R} t\tau$. ∎

We analyse peaks of the form $\twoheadleftarrow \cdot \twoheadrightarrow$. According to the definition of $\twoheadrightarrow$, they fall into the three cases: (a) ${}^*_\mathcal{S}\!\leftarrow \cdot \to^*_\mathcal{S}$, (b) ${}_{\mathcal{R}_\mathcal{S}}\!\leftarrow \cdot \to^*_\mathcal{S}$, and (c) ${}_{\mathcal{R}_\mathcal{S}}\!\leftarrow \cdot \to_{\mathcal{R}_\mathcal{S}}$. For case (a) we can apply confluence of $\mathcal{S}$ to show decreasingness of the peak. The remaining cases are more complicated. We start with a localized version of (b). In the next Lemmata 6.2.6, 6.2.9, and 6.2.10 we assume $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$ and confluence of $\mathcal{S}$.

Lemma 6.2.6. *If $t\ {}_{\mathcal{R}_\mathcal{S}}\!\leftarrow s \to_\mathcal{S} u$ then $t \to^*_{\mathcal{R}\cup\mathcal{S}} \cdot\ {}^*_{\mathcal{R}_\mathcal{S}}\!\leftarrow u$.*

Proof. We perform induction on s. Suppose $t\ {}_{\mathcal{R}_\mathcal{S}}\!\xleftarrow{p} s \xrightarrow{q}_\mathcal{S} u$. By the definition of $\mathcal{R}_\mathcal{S}$ we may assume $\ell_1\rho \to r_1 \in \mathcal{R}$ for some linear term ℓ_1 and $\rho : \mathcal{V} \to \mathcal{V}$, $s|_p = \ell_1\sigma$, $t|_p = r_1\tau$, and $\sigma \to^*_\mathcal{S} \rho\tau$, as well as $\ell_2 \to r_2 \in \mathcal{S}$, $s|_q = \ell_2\mu$, and $u|_q = r_2\mu$. Due to $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$, neither $p \backslash q \in \mathcal{P}\mathsf{os}_\mathcal{F}(\ell_2)$ nor $q \backslash p \in \mathcal{P}\mathsf{os}_\mathcal{F}(\ell_1)$ holds. We distinguish several cases concerning the relation of p and q.

- Suppose $p = \varepsilon$. Then there is a variable position q_1 of x_1 in ℓ_1 with $q_1 \leqslant q$. Since $x_1\rho\tau \overset{*}{\mathcal{S}}\!\leftarrow x_1\sigma \to_{\mathcal{S}} u|_{q_1}$ holds, we have $x_1\rho\tau \to^*_{\mathcal{S}} v \overset{*}{\mathcal{S}}\!\leftarrow u|_{q_1}$ for some v by confluence of $\mathcal{S}$. We define the substitutions μ_1 and ν as follows:

$$\mu_1(x) = \begin{cases} u|_{q_1} & \text{if } x = x_1 \\ x\sigma & \text{otherwise} \end{cases} \qquad \nu(x) = \begin{cases} v & \text{if } x = x_1\rho \\ x\tau & \text{otherwise} \end{cases}$$

  We have $\tau \to^*_{\mathcal{S}} \nu$, and also $u = \ell_1\mu_1$ by linearity of ℓ_1. Moreover, $\mu_1 \to^*_{\mathcal{S}} \rho\nu$ because $x\mu_1 \to^*_{\mathcal{S}} v = x_1\rho\nu = x\rho\nu$ if $x = x_1$, and $x\mu = x\sigma \to^*_{\mathcal{S}} x\nu$ otherwise. Therefore, we obtain $t = r_1\tau \to^*_{\mathcal{S}} r_1\nu \; {}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow \ell_1\mu_1 = u$.

- Suppose $q = \varepsilon$. We may presume $\mathcal{V}\mathsf{ar}(\ell_1) \cap \mathcal{V}\mathsf{ar}(\ell_2) = \varnothing$, and thus $\sigma = \mu$ can be assumed. Since $\ell_2\sigma \to_{\mathcal{R}_{\mathcal{S}}} t$ holds, by Lemma 6.2.5 we obtain $t \to^*_{\mathcal{R}_{\mathcal{S}}} \ell_2\nu$ for some ν with $\sigma \to^{=}_{\mathcal{R}_{\mathcal{S}}} \nu$. Thus, $t \to^*_{\mathcal{R}_{\mathcal{S}}} \ell_2\nu \to_{\mathcal{S}} r_2\nu \; {}^*_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow r_2\sigma = r_2\mu = u$.

- If $p = ip'$ and $q = jq'$ for some $i, j \in \mathbb{N}$ with $i \neq j$, one can easily verify $t \overset{q}{\to}_{\mathcal{S}} \cdot \; {}_{\mathcal{R}_{\mathcal{S}}}\!\overset{p}{\leftarrow} u$.

- Otherwise, $p = ip'$ and $q = iq'$ for some $i \in \mathbb{N}$. Since $t|_i \; {}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow s|_i \to_{\mathcal{S}} u|_i$ holds, the induction hypothesis yields $t|_i \to^*_{\mathcal{R}\cup\mathcal{S}} \cdot \; {}^*_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow u|_i$. Therefore $t \to^*_{\mathcal{R}\cup\mathcal{S}} \cdot \; {}^*_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow u$.

∎

In order to handle peaks of shape ${}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow \cdot \to^*_{\mathcal{S}}$ we show auxiliary lemmata for ARSs. In the next two lemmata $\to$ stands for $\to_1 \cup \to_2$ and $>$ for $(\to_1/\to_2)^+$, and we write $b \to_a c$ if $a \to^* b \to c$.

Lemma 6.2.7. *The next two properties hold:*

(a) For all a, b, c, d with $a > b$, if $b \overset{\vee}{\longleftrightarrow}{}^*_a c \to^* d$ then $b \overset{\vee}{\longleftrightarrow}{}^*_a d$.

(b) For all a, b, c with $a \to^* b \to^*_1 c$ there exists some d such that $b \to^{=}_1 d \overset{\vee}{\to}{}^*_a c$.

Proof. (a) We distinguish three cases: If $b = c$, then each step of $b \to^* d$ can be labelled by b itself. Since $a > b$ was assumed, we have $b \overset{\vee}{\longleftrightarrow}{}^*_a d$. If $b \neq c$ then there exists some e and label a' with $a > a'$ such that either $b \overset{\vee}{\longleftrightarrow}{}^*_a e \to_{a'} c \to^* d$ or $b \overset{\vee}{\longleftrightarrow}{}^*_a e \; {}_{a'}\!\leftarrow c \to^* d$. In both cases we have $a' \to^* c$, and thus again all steps of $c \to^* d$ can be labelled with a' to obtain $b \overset{\vee}{\longleftrightarrow}{}^*_a d$. (b) First note that the claim trivially holds if $b \to^{=}_1 c$. Otherwise we distinguish whether $a = b$. If $a \neq b$, then we can label each step of $b \to^*_1 c$ by b itself and obtain $b \overset{\vee}{\to}{}^*_a c$. Otherwise there exists some d with $b \to_1 d \to^* c$. We can label each step of $d \to^* c$ by d. Since to $a \to^+_1 b$ and $b \to_1 d$, we have $b \to_1 d \overset{\vee}{\to}{}^*_a c$. ∎

Lemma 6.2.8. *Let ${}_1\!\leftarrow \cdot \to_2 \subseteq \to^* \cdot \; {}^*_1\!\leftarrow$. If $b \; {}_1\!\leftarrow a \to^*_2 c$ then $b \overset{\vee}{\longleftrightarrow}{}^*_a \cdot \; {}^{=}_1\!\leftarrow c$.*

Proof. Let $b \;{}_1\!\leftarrow a \rightarrow_2^n c$. We show the claim by induction on n. If $n = 0$ then trivially the claim holds. Otherwise, $a \rightarrow_2^{n-1} d \rightarrow_2 c$ for some d. The induction hypothesis yields $b \overset{\vee}{\leftrightarrow}_a^* e \;\overset{=}{}_1\!\leftarrow d$ for some e. We distinguish two cases.

- If $d = e$ then $b \overset{\vee}{\leftrightarrow}_a^* e = d \rightarrow_1 c$. Thus $b \overset{\vee}{\leftrightarrow}_a^* c$ by (1).
- Suppose $d \rightarrow_1 e$. Because we have $e \;{}_1\!\leftarrow d \rightarrow_2 c$, by the assumption $e \rightarrow^* f \;{}_1^*\!\leftarrow c$ for some f. Since $a \rightarrow_2^* d \rightarrow_1 e \rightarrow^* f$ holds, we obtain $e \overset{\vee}{\rightarrow}_a^* f$ by (2). Moreover, $c \rightarrow_1^* f$ implies $c \rightarrow_1^{=} \cdot \overset{\vee}{\rightarrow}_a^* f$ by (2). Hence, $b \overset{\vee}{\leftrightarrow}_a^* \cdot \;\overset{=}{}_1\!\leftarrow c$.

∎

Lemma 6.2.9. *If* $t \;{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow s \rightarrow_\mathcal{S}^* u$ *then* $t \overset{\vee}{\leftrightarrow}_s^* \cdot \;\overset{=}{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow u$.

Proof. By Lemma 6.2.6 we have that $t \;{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow s \rightarrow_\mathcal{S} u$ implies $t \rightarrow_{\mathcal{R}\cup\mathcal{S}}^* \cdot \;{}_{\mathcal{R}_\mathcal{S}}^*\!\leftarrow u$. The claim follows by instantiating Lemma 6.2.8 with $\rightarrow_1$ as $\rightarrow_{\mathcal{R}_\mathcal{S}}$ and $\rightarrow_2$ as $\rightarrow_\mathcal{S}$. ∎

Lastly, peaks of case (c), of shape ${}_{\mathcal{R}_\mathcal{S}}\!\leftarrow \cdot \rightarrow_{\mathcal{R}_\mathcal{S}}$, are considered.

Lemma 6.2.10. *If* $t \;{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow s \rightarrow_{\mathcal{R}_\mathcal{S}} u$ *then* $t \overset{\vee}{\leftrightarrow}_s^* u$ *or* $t \rightarrow_\mathcal{S}^* \cdot \leftrightarrow_{\mathsf{CP}_\mathcal{S}(\mathcal{R})} \cdot \;{}_\mathcal{S}^*\!\leftarrow u$.

Proof. We perform induction on s. Suppose $t \;{}_{\mathcal{R}_\mathcal{S}}\!\overset{p}{\leftarrow} s \overset{q}{\rightarrow}_{\mathcal{R}_\mathcal{S}} u$. By the definition of $\mathcal{R}_\mathcal{S}$ we can assume $\ell_1\rho_1 \to r_1, \ell_2\rho_2 \to r_2 \in \mathcal{R}$ for some linear terms ℓ_1, ℓ_2 and $\rho_1, \rho_2 : \mathcal{V} \to \mathcal{V}$, and

$$\begin{array}{lll} s|_p = \ell_1\sigma_1 & t|_p = r_1\tau_1 & \sigma_1 \rightarrow_\mathcal{S}^* \rho_1\tau_1 \\ s|_q = \ell_2\sigma_2 & u|_q = r_2\tau_2 & \sigma_2 \rightarrow_\mathcal{S}^* \rho_2\tau_2 \end{array}$$

Except for symmetric cases, the relation of p and q falls into the next four cases:

- Suppose $q = \varepsilon$, and $p \in \mathcal{P}\mathsf{os}_\mathcal{F}(\ell_2)$. We have $\ell_1\rho_1\tau_1 \;{}_\mathcal{S}^*\!\leftarrow s \rightarrow_\mathcal{S}^* \ell_2|_p\rho_2\tau_2$. Without loss of generality $\mathcal{V}\mathsf{ar}(\ell_1\rho_1) \cap \mathcal{V}\mathsf{ar}(\ell_2\rho_2) = \varnothing$, and thus we may assume $\tau = \tau_1 \cup \tau_2$ is a well-defined substitution. The substitution τ is an $\mathcal{S}$-unifier of $\ell_1\rho_1$ and $\ell_2\rho_2|_p$. Because $x\sigma_2 \rightarrow_\mathcal{S}^* x\rho_2\tau$ holds for all $x \in \mathcal{V}\mathsf{ar}(\ell_2)$,
$$t = (\ell_2\sigma_2)[r_1\tau]_p \rightarrow_\mathcal{S}^* (\ell_2\rho_2\tau)[r_1\tau]_p \leftrightarrow_{\mathsf{CP}_\mathcal{S}(\mathcal{R})} r_2\tau = u$$
- Suppose $q = \varepsilon$, and $p \notin \mathcal{P}\mathsf{os}_\mathcal{F}(\ell_2)$ and p_2 is a variable occurrence of x_2 in ℓ_2 with $p_2 \leqslant p$. Since $t|_{p_2} \;{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow x_2\sigma_2 \rightarrow_\mathcal{S}^* x_2\rho_2\tau_2$, Lemma 6.2.9 yields $t|_{p_2} \overset{\vee}{\leftrightarrow}_{s|_{p_2}}^* v \;\overset{=}{}_{\mathcal{R}_\mathcal{S}}\!\leftarrow x_2\rho_2\tau_2$ for some v. Because $s = \ell_2\sigma_2$, $t|_{p_2} \overset{\vee}{\leftrightarrow}_{s|_{p_2}}^* v$, and $\sigma_2 \rightarrow_\mathcal{S}^* \rho_2\tau_2$ hold, by closure under contexts of rewrite relations and $>$ we obtain
$$t = (\ell_2\sigma_2)[t|_{p_2}]_{p_2} \overset{\vee}{\leftrightarrow}_s^* (\ell_2\sigma_2)[v]_{p_2} \rightarrow_\mathcal{S}^* (\ell_2\rho_2\tau_2)[v]_{p_2}$$
Thus, $t \overset{\vee}{\leftrightarrow}_s^* (\ell_2\rho_2\tau_2)[v]_{p_2}$. Since $x_2\rho_2\tau_2 \rightarrow_{\mathcal{R}_\mathcal{S}}^{=} v$ holds and ℓ_2 is linear,
$$\ell_2\rho_2\tau_2 \rightarrow_{\mathcal{R}_\mathcal{S}}^{=} (\ell_2\rho_2\tau_2)[v]_{p_2}$$

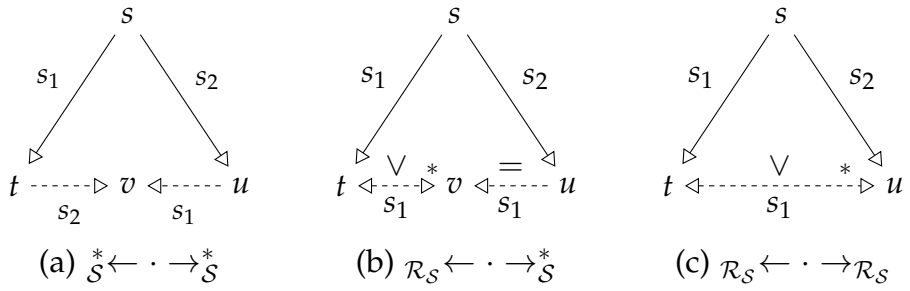

Figure 6.2: Decreasingness of $\rightarrowtriangle$.

is deduced. Here we distinguish two cases. If $\ell_2\rho_2\tau_2 = (\ell_2\rho_2\tau_2)[v]_{p_2}$, we obtain

$$t \overset{\vee}{\leftarrowtriangle\rightarrow}^*_s \ell_2\rho_2\tau_2 \rightarrow_{\mathcal{R}} u$$

Otherwise, $\ell_2\rho_2\tau_2 \rightarrow_{\mathcal{R}_{\mathcal{S}}} (\ell_2\rho_2\tau_2)[v]_{p_2}$. Since by Lemma 6.2.5 there exists ν with $\tau_2 \rightarrow^{=}_{\mathcal{R}_{\mathcal{S}}} \nu$ such that $(\ell_2\rho_2\tau_2)[v]_{p_2} \rightarrow^*_{\mathcal{R}_{\mathcal{S}}} \ell_2\rho_2\nu$, finally we obtain

$$t \overset{\vee}{\leftarrowtriangle\rightarrow}^*_s (\ell_2\rho_2\tau_2)[v]_{p_2} \rightarrow^*_{\mathcal{R}_{\mathcal{S}}} \ell_2\rho_2\nu \rightarrow_{\mathcal{R}} r_2\nu \ {}^*_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow r_2\tau_2 = u$$

Because $s > t$ and $s > u$ hold, in both cases $t \overset{\vee}{\leftarrowtriangle\rightarrow}^*_s u$ is concluded.

- If $p = ip'$ and $q = jq'$ for some $i, j \in \mathbb{N}$ with $i \neq j$, one can easily verify $t \overset{q}{\rightarrow}_{\mathcal{R}_{\mathcal{S}}} \cdot {}_{\mathcal{R}_{\mathcal{S}}}\overset{p}{\leftarrow} u$, which implies $t \overset{\vee}{\leftrightarrow}^*_s u$.
- Otherwise, $p = ip'$ and $q = iq'$ for some $i \in \mathbb{N}$. Since $t|_i \ {}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow s|_i \rightarrow_{\mathcal{R}_{\mathcal{S}}} u|_i$ holds, by the induction hypothesis $t|_i \overset{\vee}{\leftrightarrow}^*_{s|_i} u|_i$ or $t|_i \rightarrow^*_{\mathcal{S}} \cdot \leftrightarrow_{\mathsf{CP}_{\mathcal{S}}(\mathcal{R})} \cdot \overset{*}{\mathcal{S}}\leftarrow u|_i$ is deduced. Thus, $t \overset{\vee}{\leftrightarrow}^*_s u$ or $t \rightarrow^*_{\mathcal{S}} \cdot \leftrightarrow_{\mathsf{CP}_{\mathcal{S}}(\mathcal{R})} \cdot \overset{*}{\mathcal{S}}\leftarrow u$ is concluded.

∎

Now we are ready to prove the main theorem.

Proof of Theorem 6.1.6. Suppose that $\mathcal{S}$ is a confluent TRS, $\mathcal{R}/\mathcal{S}$ is terminating, and $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$. We show that $\mathcal{R} \cup \mathcal{S}$ is confluent if and only if $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$. Since the "only if"-direction is trivial, we only show the "if"-direction. Assume $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$. Because confluence of $\rightarrowtriangle$ implies confluence of $\mathcal{R} \cup \mathcal{S}$, according to Theorem 6.2.2 it is enough to show decreasingness of $\rightarrowtriangle$. Let $t \ {}_{s_1}\!\leftarrowtriangle s \rightarrowtriangle_{s_2} u$. As mentioned, following the definition of $\rightarrowtriangle$, we distinguish three cases.

(a) If $t \ \overset{*}{\mathcal{S}}\leftarrow s \rightarrow^*_{\mathcal{S}} u$ then $t \rightarrow^*_{\mathcal{S}} v \ \overset{*}{\mathcal{S}}\leftarrow u$ for some v by confluence of $\mathcal{S}$.

(b) If $t \ {}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow s \rightarrow^*_{\mathcal{S}} u$ then $t \overset{\vee}{\leftarrowtriangle\rightarrow}^*_s v \ {}^{=}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow u$ for some v by Lemma 6.2.9.

(c) If $t \ {}_{\mathcal{R}_{\mathcal{S}}}\!\leftarrow s \rightarrow_{\mathcal{R}_{\mathcal{S}}} u$ then $t \overset{\vee}{\leftarrowtriangle\rightarrow}^*_s u$ for some v by Lemma 6.2.10 and joinability of $\mathcal{S}$-critical pairs.

In all cases decreasingness is established, as seen in Figure 6.2. ∎

6.3 Joinability of Extended Critical Pairs

The biggest challenge in applying Theorem 6.1.6 is to check $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$ automatically. The problem is that $\mathsf{CP}_{\mathcal{S}}(\mathcal{R})$ may contain infinitely many pairs.

Before proceeding, we extend the notions of unifiability and unifiers from Chapter 2. Let $\mathcal{E}$ and $\mathcal{S}$ be sets of equations, and X the set of all variables in $\mathcal{E}$. Note that a TRS can be taken for $\mathcal{S}$. An *$\mathcal{S}$-unification problem* is a finite set of equalities $\{s_1 \approx t_1, \ldots, s_n \approx t_n\}$. Given a substitution σ, we write $\mathcal{E}\sigma$ for $\{s\sigma \approx t\sigma \mid s \approx t \in \mathcal{E}\}$.

Definition 6.3.1. *Let $\mathcal{E}$ be an $\mathcal{S}$-unification problem. An* $\mathcal{S}$-unifier *of $\mathcal{E}$ is a substitution σ such that $\mathcal{E}\sigma \subseteq \leftrightarrow_{\mathcal{S}}^*$.*

A substitution σ is *more general* than a substitution σ' on X ($\sigma \lesssim_{\mathcal{S}}^{X} \sigma'$), if there exists a substitution τ such that $x\sigma' \leftrightarrow_{\mathcal{S}}^* x\sigma\tau$ for all $x \in X$.

Definition 6.3.2. *Let $\mathcal{U}$ be a set of $\mathcal{S}$-unifiers of $\mathcal{E}$.*

1. *$\mathcal{U}$ is* complete *if for every $\mathcal{S}$-unifier of $\mathcal{E}$ there is a more general element in $\mathcal{U}$, and*
2. *if in addition all elements in $\mathcal{U}$ are minimal with respect to $\lesssim_{\mathcal{S}}^{X}$, we call $\mathcal{U}$* minimal complete.
3. *A substitution σ is an* $\mathcal{S}$-most general unifier *($\mathcal{S}$-*mgu*) of $\mathcal{E}$, if $\{\sigma\}$ is a minimal complete set of $\mathcal{S}$-unifiers of $\mathcal{E}$.*

The special case of $\mathcal{S} = \varnothing$ corresponds to syntactic unification, unifiers and mgu's as defined in Chapter 2.

A set of equalities $\mathcal{E} = \{x_1 \approx t_1, \ldots, x_n \approx t_n\}$ is in *solved form*, if the x_i are pairwise distinct variables, and none of the x_i occurs in any t_j. For $\mathcal{E}$ in solved form, we write $\overrightarrow{\mathcal{E}}$ for the induced substitution $\{x_1 \mapsto t_1, \ldots, x_n \mapsto t_n\}$. Note that in contrast to syntactic unifiability ($\mathcal{S} = \varnothing$), $\mathcal{S}$-unifiability does not ensure the existence of an $\mathcal{S}$-mgu in general.

The standard approach to test $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$ is to compute a minimal complete set of $\mathcal{S}$-unifiers for ℓ_1 and $\ell_2|_p$ for each combination of rules $\ell_1 \to r_1$, $\ell_2 \to r_2$ and a position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$. Then, joinability of its induced critical pairs ensures joinability for all $\mathcal{S}$-unifiers. However, depending on $\mathcal{S}$, the computation of minimal complete sets varies, and worse, minimal complete sets may not even exist for $\mathcal{S}$-unifiable terms. In this section we give sufficient conditions for the joinability and non-joinability of $\mathcal{S}$-critical pairs without performing specific equational unification algorithms.

For the first we show that a most general unifier of strongly $\mathcal{S}$-stable terms is always a most general $\mathcal{S}$-unifier. As the next lemma shows, this allows us to compute $\mathcal{S}$-critical pairs by means of syntactic unification. Here a term t

is *strongly $\mathcal{S}$-stable* if for every position $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(t)$ there are no term u and substitution σ such that $t|_p\sigma \to_{\mathcal{S}}^* \cdot \xrightarrow{\varepsilon}_{\mathcal{S}} u$. Note that $t\sigma$ is strongly $\mathcal{S}$-stable if t and $x\sigma$ are strongly $\mathcal{S}$-stable for all variables x in t.

Lemma 6.3.3. *Let t be a linear term. Then (a) implies (b) where*

(a) for every σ we have that $t\sigma \xrightarrow{p}_{\mathcal{S}} u$ implies $p \in \mathcal{P}\mathsf{os}_{\mathcal{V}}(t)$,

(b) for every σ we have that $t\sigma \to_{\mathcal{S}}^* u$ implies $u = t\tau$ for some τ.

Proof. By induction on the length of $t\sigma \to_{\mathcal{S}}^* u$. ∎

Lemma 6.3.4. *If $\mathsf{SNO}(\mathcal{R},\mathcal{S})$ then ℓ is strongly $\mathcal{S}$-stable for all $\ell \to r \in \mathcal{R}$.*

Proof. Suppose $\mathsf{SNO}(\mathcal{R},\mathcal{S})$ and let $\ell \to r$ be a rule in $\mathcal{R}$ and $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell)$. We write t for $\mathsf{REN}(\ell|_p)$. Clearly, t is linear and by $\mathsf{SNO}(\mathcal{R},\mathcal{S})$ condition (a) of Lemma 6.3.3 holds for t. Thus the lemma implies that condition (b) also holds. Now consider the reduction $\ell|_p\sigma \to_{\mathcal{S}}^* u \xrightarrow{p}_{\mathcal{S}} v$. It is not difficult to see that there exists a substitution ρ with $t\rho = \ell|_p\sigma$. Then by condition (b) of Lemma 6.3.3 we have $u = t\tau$ for some τ. By condition (a) the position p must be a variable position, and thus $p \neq \varepsilon$. So $\ell|_p\sigma \to_{\mathcal{S}}^* v \xrightarrow{\varepsilon} u$ does not hold. Hence ℓ must be strongly $\mathcal{S}$-stable. ∎

In order to show the claim on mgu's, we recall the standard inference rules for syntactic unification from [10]. These rules are defined over sets of equalities on terms.

$$\textsc{Eliminate}\quad \frac{\{x \approx t\} \uplus \mathcal{E}}{\{x \approx t\} \cup \mathcal{E}\{x \mapsto t\}} \qquad \text{if } x \notin \mathcal{V}\mathsf{ar}(t)$$

$$\textsc{Orient}\quad \frac{\{t \approx x\} \uplus \mathcal{E}}{\{x \approx t\} \cup \mathcal{E}} \qquad \text{if } t \notin \mathcal{V}$$

$$\textsc{Delete}\quad \frac{\{t \approx t\} \uplus \mathcal{E}}{\mathcal{E}}$$

$$\textsc{Decompose}\quad \frac{\{f(s_1,\ldots,s_n) \approx f(t_1,\ldots,t_n)\} \uplus \mathcal{E}}{\{s_1 \approx t_1,\ldots,s_n \approx t_n\} \cup \mathcal{E}}$$

We write $\Longrightarrow$ for a derivation by the inference rules. The following lemma states that a most general unifier can be computed by a sequence of derivations.

Lemma 6.3.5 ([10])**.** *If s and t are unifiable, there exists $\mathcal{E}$ in solved form such that $\{s \approx t\} \Longrightarrow^* \mathcal{E}$ and $\overrightarrow{\mathcal{E}}$ is an mgu of s and t.* ∎

The next lemma shows that the inference rules of syntactic unification preserve strong $\mathcal{S}$-stability and $\mathcal{S}$-unifiability. We say that a set $\mathcal{E}$ of equalities is strongly $\mathcal{S}$-stable if s and t are strongly $\mathcal{S}$-stable for all $s \approx t \in \mathcal{E}$.

Lemma 6.3.6. *Let $\mathcal{S}$ be a confluent TRS. If $\mathcal{E}_1$ is strongly $\mathcal{S}$-stable, $\mathcal{E}_1\sigma \subseteq \downarrow_{\mathcal{S}}$, and $\mathcal{E}_1 \Longrightarrow \mathcal{E}_2$, then $\mathcal{E}_2\sigma \subseteq \downarrow_{\mathcal{S}}$ and $\mathcal{E}_2$ is strongly $\mathcal{S}$-stable.*

Proof. Suppose $\mathcal{E}_1$ is strongly $\mathcal{S}$-stable, $\mathcal{E}_1\sigma \subseteq \downarrow_{\mathcal{S}}$, and $\mathcal{E}_1 \Longrightarrow \mathcal{E}_2$. We distinguish the inference of $\mathcal{E}_1 \Longrightarrow \mathcal{E}_2$. Because the cases of Delete and Orient are trivial, below we only consider the other two cases:

- Eliminate: Suppose $\mathcal{E}_1 = \{x \approx t\} \uplus \mathcal{E}'$ and $\mathcal{E}_2 = \{x \approx t\} \cup \mathcal{E}'\mu$, where $\mu = \{x \mapsto t\}$ and $x \notin \mathcal{V}\mathsf{ar}(t)$. We claim $\mu\sigma \leftrightarrow^*_{\mathcal{S}} \sigma$. Actually it follows from the assumption $x\sigma \downarrow_{\mathcal{S}} t\sigma$. We now prove $\mathcal{E}_2\sigma \subseteq \downarrow_{\mathcal{S}}$. It is sufficient to show $u\mu\sigma \downarrow_{\mathcal{S}} v\mu\sigma$ for an arbitrary $u \approx v \in \mathcal{E}'$. Because $u\sigma \downarrow_{\mathcal{S}} v\sigma$ by assumption, the claim yields $u\mu\sigma \leftrightarrow^*_{\mathcal{S}} v\mu\sigma$. Therefore $u\mu\sigma \downarrow_{\mathcal{S}} v\mu\sigma$ is concluded from confluence of $\mathcal{S}$. To show strong $\mathcal{S}$-stability of $\mathcal{E}_2$, fix $u \approx v \in \mathcal{E}'$. Since u, v, and $x\mu$ are strongly $\mathcal{S}$-stable, so are $u\mu$ and $v\mu$.

- Decompose: Suppose $\mathcal{E}_1 = \{s \approx t\} \uplus \mathcal{E}'$ and $\mathcal{E}_2 = \{s_1 \approx t_1, \ldots, s_n \approx t_n\} \cup \mathcal{E}'$ with $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$. Since $\mathcal{E}$ is strongly $\mathcal{S}$-stable, and thus s and t are, s_i and t_i are also strongly $\mathcal{S}$-stable for all $1 \leqslant i \leqslant n$. Furthermore, due to strong $\mathcal{S}$-stability of s and t, $s\sigma \downarrow_{\mathcal{S}} t\sigma$ implies $s_i\sigma \downarrow_{\mathcal{S}} t_i\sigma$ for all $1 \leqslant i \leqslant n$. Therefore, the claim holds.

∎

We arrive at the aforementioned sufficient condition.

Theorem 6.3.7. *Let $\mathcal{S}$ be a confluent TRS. An mgu of strongly $\mathcal{S}$-stable terms s and t is an $\mathcal{S}$-mgu of s and t.*

Proof. Let μ be an arbitrary mgu of strongly $\mathcal{S}$-stable terms s and t. Since μ is trivially an $\mathcal{S}$-unifier of s and t, it is enough to show that μ is more general than an arbitrary $\mathcal{S}$-unifier σ of s and t. By using Lemma 6.3.5 there is an $\mathcal{E}$ in solved form such that $\{s \approx t\} \Longrightarrow^* \mathcal{E}$ and $\overrightarrow{\mathcal{E}}$ is an mgu of s and t. Because $s\sigma \leftrightarrow^*_{\mathcal{S}} t\sigma$ and $\mathcal{S}$ is confluent, we have $\{s \approx t\}\sigma \subseteq \downarrow_{\mathcal{S}}$, and thus $\mathcal{E}\sigma \subseteq \downarrow_{\mathcal{S}}$ is obtained by induction on the length of $\Longrightarrow^*$ using Lemma 6.3.6. Since $\mathcal{E}$ is in solved form, $x\sigma \downarrow_{\mathcal{S}} x\overrightarrow{\mathcal{E}}\sigma$ holds for all $x \in \mathcal{D}\mathsf{om}(\overrightarrow{\mathcal{E}})$. This means $\sigma \leftrightarrow^*_{\mathcal{S}} \overrightarrow{\mathcal{E}}\sigma$. Since μ is an mgu, there is a substitution ρ with $\overrightarrow{\mathcal{E}} = \mu\rho$. Thus $\sigma \leftrightarrow^*_{\mathcal{S}} \mu\rho\sigma$. Hence μ is more general than σ. ∎

When automating Theorem 6.1.6, confluence of $\mathcal{S}$ and $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$ can be assumed. Therefore, according to Theorem 6.3.7 and Lemma 6.3.4, a syntactical overlap by an mgu μ is also an $\mathcal{S}$-overlap by $\mathcal{S}$-mgu μ. Thus joinability of its syntactical critical pairs implies joinability of $\mathcal{S}$-critical pairs induced by any $\mathcal{S}$-unifier.

Example 6.3.8 (continued from Example 6.1.9)**.** *We consider again the example with* $\mathcal{R} = \{1, 2\}$ *and* $\mathcal{S} = \{3, 4\}$*. Take the first and second rules renamed:*

$$1\colon\ \mathsf{eq}(\mathsf{s}(n), x : xs, x : ys) \to \mathsf{eq}(n, xs, ys) \qquad 2\colon\ \mathsf{eq}(m, zs, zs) \to \mathsf{T}$$

We know that there is an overlap between 1 *and* 2 *at root position with mgu*

$$\mu = \{m \mapsto \mathsf{s}(n),\ zs \mapsto x : xs,\ ys \mapsto xs\}.$$

Elsewhere, even $\mathcal{S}$*-overlaps cannot occur. The induced critical pair* $(\mathsf{eq}(n, xs, xs), \mathsf{T})$ *is trivially joinable by the second rule. Hence* $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$ *holds.*

Confluence of $\mathcal{S}$ cannot be dropped in Theorem 6.3.7.

Example 6.3.9. *Consider the TRS* $\mathcal{S}$

$$\mathsf{g}(x, y) \to \mathsf{f}(x, x) \qquad \mathsf{g}(x, y) \to \mathsf{f}(x, y)$$

The terms $\mathsf{f}(x_1, x_1)$ *and* $\mathsf{f}(x, y)$ *are both strongly* $\mathcal{S}$*-stable, and the substitution*

$$\mu = \{x \mapsto x_1, y \mapsto x_1\}$$

is a most general unifier. However, μ *is not an* $\mathcal{S}$*-mgu, because* μ *is not more general than the other* $\mathcal{S}$*-unifier* $\{x_1 \mapsto x\}$*.*

Unjoinability of $\mathcal{S}$-critical pairs can be tested in a similar way to checking non-confluence of a TRS with the function TCAP ([86]).

Definition 6.3.10 ([29])**.** *Let* t *be a term, and* $\mathcal{R}$ *a TRS. We define* $\mathsf{TCAP}_{\mathcal{R}}(t)$ *inductively as a fresh variable, when* t *is a variable or when* $t = f(t_1, \ldots, t_n)$ *and* ℓ *and* u *unify for some (renamed) rule* $\ell \to r \in \mathcal{R}$*, and* u*, otherwise. Here* u *stands for* $f(\mathsf{TCAP}_{\mathcal{R}}(t_1), \ldots, \mathsf{TCAP}_{\mathcal{R}}(t_n))$*.*

Lemma 6.3.11. *Let* $\ell_1 \to r_1$*,* $\ell_2 \to r_2 \in \mathcal{R}$ *and* $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$*. If* $\ell_1\sigma \leftrightarrow^*_{\mathcal{R}} \ell_2|_p\sigma$*, and* $\mathsf{TCAP}_{\mathcal{R}}(r_2)$ *and* $\mathsf{TCAP}_{\mathcal{R}}(\ell_2[r_1]_p)$ *do not unify,* $\mathcal{R}$ *is not confluent.*

Proof. Using the fact that if $s\sigma \downarrow_{\mathcal{R}} t\tau$ then $\mathsf{TCAP}_{\mathcal{R}}(s)$ and $\mathsf{TCAP}_{\mathcal{R}}(t)$ must unify, see [86]. ∎

Example 6.3.12 (continued from Example 6.0.1)**.** *Recall* $\mathcal{R} = \{1, 2\}$ *and* $\mathcal{S} = \{3\}$*:*

$$1\colon\ \mathsf{f}(x, x) \to \mathsf{a} \qquad 2\colon\ \mathsf{f}(y, \mathsf{g}(y)) \to \mathsf{b} \qquad 3\colon\ \mathsf{c} \to \mathsf{g}(\mathsf{c})$$

where variables are renamed in rule 2. We denote i-th rule by $\ell_i \to r_i$*. While* ℓ_1 *and* $\ell_2|_\varepsilon$ *are* $(\mathcal{R} \cup \mathcal{S})$*-unifiable with* $\{x, y \mapsto \mathsf{c}\}$*, we have*

$$\mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(\ell_2[r_1]_\varepsilon) = \mathsf{a} \quad \textit{and} \quad \mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(r_2) = \mathsf{b}$$

which do not unify. Thus, by Lemma 6.3.11, $\mathcal{R} \cup \mathcal{S}$ *is not confluent.*

In automation we need to test $\mathcal{S}$-unifiability of ℓ_1 and $\ell_2|_p$. This can be automated by first-order theorem provers for unit equational problems, so-called UEQ. We briefly describe the approach. Roughly summarized, such theorem provers work as follows. The input is a set of equations $\mathcal{E}$ and *goal* φ containing arbitrary quantifiers

$$Q_1 x_1, \ldots, Q_n x_n \; s \approx t$$

where Q_i stands for either $\forall$ or $\exists$ and $x_1, \ldots, x_n$ are the variables appearing in s and t. The theorem prover tries to answer whether $\mathcal{E} \models \varphi$. The approach is to first negate and skolemize φ which results in a formula $s' \not\approx t'$. Note that if we have $Q_i = \forall$ for all $1 \leq i \leq n$ then both s' and t' are ground terms. If s' and t' are $\mathcal{E}$-unifiable then the goal is inconsistent with the axioms and thus we have $\mathcal{E} \models \varphi$. Most theorem provers for UEQ are based on a variant of completion called unfailing completion [12] which is *refutationally complete*. This means that if $\mathcal{E} \models \varphi$ then unifiability will (theoretically) be detected after finitely many time and thus we are guaranteed that the conjecture will be proven.

Testing $\mathcal{S}$-unifiability of ℓ_1 and $\ell_2|_p$ means asking whether $\mathcal{S} \models \varphi$ where φ is

$$\exists x_1, \ldots, \exists x_n \; \ell_1 \approx \ell_2|_p$$

and $x_1, \ldots, x_n$ are the variables of ℓ_1 and $\ell_2|_p$. If we input φ together with $\mathcal{S}$, then the theorem prover will check $\mathcal{S}$-unifiability of ℓ_1 and $\ell_2|_p$ which is precisely our original question. If unifiability is proven then we can employ TCAP and Lemma 6.3.11. Depending on the power of the prover, sometimes invalid conjecture can be refuted, i.e. it can be shown that

$$\mathcal{S} \not\models \exists x_1, \ldots, \exists x_n \; \ell_1 \approx \ell_2|_p.$$

Then we can conclude the absence of $\mathcal{S}$-unifiers. However in this case *refutational completeness*[2] does not hold, i.e. the theorem prover is not (theoretically) guaranteed to refute the conjecture after finitely many time.

Note that in contrast to [86], Lemma 6.3.11 only requires to know that $\mathcal{S}$-unifiability holds, but it does not require the computation of $\mathcal{S}$-unifiers. In fact most theorem provers test only unifiability without computing any concrete unifier.

By the above approach indeed non-confluence of the TRS of Example 6.3.12 can be proven automatically, see Section 6.4. As a final remark, note that from the absence of a syntactic unifier we may not conclude non-existence of $\mathcal{S}$-critical pairs, as illustrated in Example 6.0.1.

[2] refutational completeness here refers to being guaranteed to show the inconsistency of the axioms w.r.t. the negated, skolemized conjecture

6.4 Experiments

In order to assess feasibility of our methods, we implemented Theorem 6.1.6 together with Theorem 6.3.7 for confluence, and Lemma 6.3.11 for non-confluence. In the next subsections we mention details of our implementation and report on experimental data.

6.4.1 Implementation

In order to automate Theorem 6.1.6 we employed TTT2 v1.06 [50] for checking relative termination $\mathcal{R}/\mathcal{S}$ and an extended version of Maxcomp [46] for testing $\mathcal{S}$-unifiability, using ordered completion. To check confluence of $\mathcal{S}$, we used the existing three state-of-the-art confluence provers: ACP v0.20 [7],[3] CSI v0.1 [86],[4] and Saigawa v1.2 [38].[5] Since termination of $\mathcal{R} \cup \mathcal{S}$ cannot be assumed, we only test joinability of $\mathcal{S}$-critical pairs by at most four rewrite steps form each side.

We give a brief overview of our procedure. Given a TRS $\mathcal{P}$, we output either YES ($\mathcal{P}$ is confluent), NO ($\mathcal{P}$ is not confluent), or MAYBE (confluence of $\mathcal{P}$ is neither proven nor disproven). We enumerate all possible partitions $\mathcal{P} = \mathcal{R} \uplus \mathcal{S}$, and then for each $(\mathcal{R}, \mathcal{S})$, we test whether $\mathsf{SNO}(\mathcal{R}, \mathcal{S})$, termination of $\mathcal{R}/\mathcal{S}$, and confluence of $\mathcal{S}$ holds. If one of these conditions does not hold, we continue with the next partition; if none is left, we return MAYBE. Otherwise, to check the last remaining condition of Theorem 6.1.6, namely $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$, we proceed in the following way: For all tuples $(\ell_1 \to r_1, p, \ell_2 \to r_2)$ where $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are rules from $\mathcal{R}$ and $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$, we test in the following order:

1. If $\mathsf{REN}(\ell_1)$ and $\mathsf{REN}(\ell_2|_p)$ are not syntactically unifiable, then no $\mathcal{S}$-overlap exists, and we continue with the next tuple. Otherwise,

2. if ℓ_1 and $\ell_2|_p$ are syntactically unifiable with σ, the current tuple forms an $\mathcal{S}$-overlap, so we test joinability of the induced critical pair.
   a) If joinability holds, we continue with the next tuple.
   b) If joinability cannot be established, we test whether $\mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(r_2\sigma)$ and $\mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(\ell_2\sigma[r_1\sigma]_p)$ syntactically unify. If they are not unifiable, return NO. Otherwise, return MAYBE

3. if ℓ_1 and $\ell_2|_p$ are not syntactically unifiable, we check $\mathcal{S} \models \ell_1 \approx \ell_2|_p$ by a theorem prover:
   a) If unsatisfiability of the formula is detected, no $\mathcal{S}$-overlap exists, and we continue.

[3] http://www.nue.riec.tohoku.ac.jp/tools/acp/
[4] http://cl-informatik.uibk.ac.at/software/csi/
[5] http://www.jaist.ac.jp/project/saigawa/

b) If satisfiability is detected, we proceed and test syntactic unifiability of $\mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(r_2)$ and $\mathsf{TCAP}_{\mathcal{R}\cup\mathcal{S}}(\ell_2[r_1]_p)$. If they are not unifiable, return NO. If they unify, return MAYBE.

c) Lastly, if the theorem prover does not provide a conclusive answer, return MAYBE

If no tuple remains, we have established $\mathsf{CP}_{\mathcal{S}}(\mathcal{R}) \subseteq \downarrow_{\mathcal{R}\cup\mathcal{S}}$ and return YES.

The procedure has the following properties. Let $1\colon\ \ell_1 \to r_1$ and $2\colon\ \ell_2 \to r_2$ be rules of $\mathcal{R}$ and $p \in \mathcal{P}\mathsf{os}_{\mathcal{F}}(\ell_2)$. Below we write ${}_1\overset{p}{\leftarrow}_{\mathcal{S}}\!\!\circ\!\!\rightarrow_2$ for the $\mathcal{S}$-critical pairs of $\ell_1 \to r_1$ and $\ell_2 \to r_2$ at p.

- If the procedure continues after step 1) or 3a) then ${}_1\overset{p}{\leftarrow}_{\mathcal{S}}\!\!\circ\!\!\rightarrow_2 = \varnothing$.
- If the procedure continues after step 2a) then ${}_1\overset{p}{\leftarrow}_{\mathcal{S}}\!\!\circ\!\!\rightarrow_2 \subseteq \downarrow_{\mathcal{R}}$ by Theorem 6.3.7 and Lemma 6.3.4.
- If the procedure outputs NO then ${}_1\overset{p}{\leftarrow}_{\mathcal{S}}\!\!\circ\!\!\rightarrow_2 \not\subseteq \downarrow_{\mathcal{R}}$ by Lemma 6.3.11.

Using these properties one can show correctness of the whole procedure using Theorem 6.1.6.

6.4.2 Experimental Results

We tested the implementation on a collection of 32 TRSs, consisting of 29 non-left-linear non-terminating TRSs in the Confluence Problem Database (Cops Nos. 1–116)[6] and Examples 6.1.7, 6.1.8 and 6.1.9. Note that Examples 6.0.1 and 6.1.11 are part of the 29 TRSs. The tests were single-threaded run on a system equipped with an Intel Core Duo L7500 with 1.6 GHz and 2 GB of RAM using a timeout of 60 seconds.

The results are depicted in Table 6.1. For more more detailed results see Appendix C. In Table 6.1, columns ACP, CSI and Saigawa show results for running the respective tools, and ACP*, CSI* and Saigawa* show results when using the respective tool to show confluence of the $\mathcal{S}$-part in Theorem 6.1.6.

It should be noted, that the criteria implemented by Saigawa apply only to left-linear systems, whereas CSI is able to show confluence of non-left-linear systems by order-sorted decomposition [25], and the implementation of ACP includes criteria based on layer preserving [59] and persistency decompositions [5], and the criterion by Gomi et al. [33].

For overall results, there are twelve TRSs for which confluence can be shown by ACP, CSI and Saigawa combined, in fact however all twelve can be shown by ACP alone. Extending with Theorem 6.1.6, there are 19 TRSs, for which confluence can

[6] http://coco.nue.riec.tohoku.ac.jp/

Table 6.1: Summary of experimental results (32 TRSs)

	ACP	ACP*	CSI	CSI*	Saigawa	Saigawa*
YES	12	**19**	7	**15**	0	**10**
NO	3	**4**	3	**3**	0	**2**
MAYBE	17	9	17	9	32	20
timeout (60 sec)	0	0	5	5	0	0

be shown by ACP*, CSI* or Saigawa* combined. Similar to the standalone-case, ACP* subsumes both other combinations. As for Example 6.1.7, 6.1.8 and 6.1.9, neither CSI, ACP nor Saigawa can show confluence, whereas all CSI*, ACP* and Saigawa* succeed. Out of the nine TRSs that ACP* missed, four TRSs (Cops Nos. 76, 77, 78, 109) contain AC rules, for which most likely the criterion in [44] applies if suitable equational unification algorithms were implemented (see Section 6.5), and five TRSs (Nos. 16, 24, 26, 27, 47) are variants of Huet's example (Example 6.0.1) or Klop's example (Example 6.1.11).

6.5 Related Work

Among others, we compare our criterion with three well-known criteria capable of proving confluence of non-left-linear and non-terminating TRSs. Note that for the second criterion below we use *reversibility* [6] for comparison, because the original criterion requires equational systems for $\mathcal{S}$ rather than rewrite systems. We say that a TRS $\mathcal{S}$ is *reversible* if ${}_{\mathcal{S}}\!\leftarrow \subseteq \rightarrow_{\mathcal{S}}^{*}$.

- **Criteria by Non-E-Overlappingness.** The criterion by Gomi et al. [31], later extended in [32] and [33], is that a *root-E-overlapping* TRS, that is also strongly *weight-preserving* or strongly *depth-preserving*, is confluent. Here E-overlaps are a generalization of overlaps, and strong non-overlappingness plays a major role in deriving decidable criteria for root-E-overlappingness.[7] A TRS is strongly depth preserving, if for any rewrite rule and any variable appearing in both sides, the minimal depth of the variable occurrences in the left-hand side is greater than or equal to the maximal depth of the right hand side's occurrences. Instead of comparing the depth of the variable

[7] $\mathcal{S}$-overlaps are sometime called $\mathcal{E}$-overlaps but should not be confused with the E-overlaps defined by Gomi et al. [31], originally introduced by Ogawa [57].

directly, one can also assign weights to function symbols and compare the weight of the variable occurrence, where the weight is the sum of the function symbols from root to its occurrence. For details of the definitions we refer to [33]. Consider the following TRS:

$$\mathsf{f}(x,x) \to \mathsf{a} \qquad \mathsf{c} \to \mathsf{g}(\mathsf{c}) \qquad \mathsf{g}(x) \to \mathsf{f}(x,x)$$

Confluence of this TRS can be established, since it is depth-preserving and root-E-overlapping. However Theorem 6.1.6 cannot be applied, since the TRS cannot be partitioned into a non-empty $\mathcal{R}$ and $\mathcal{S}$, such that $\mathcal{R}/\mathcal{S}$ is terminating — except for $\mathcal{R} = \varnothing$. On the other hand, weight-preservation and depth-preservation impose strong syntactic restrictions on the variable positions. Consider for example the TRS

$$1\colon\ \mathsf{g}(x,x) \to \mathsf{f}(x) \qquad 2\colon\ \mathsf{f}(x) \to \mathsf{f}(\mathsf{f}(x))$$

By taking $\mathcal{R} = \{1\}$ and $\mathcal{S} = \{2\}$, Theorem 6.1.6 can be applied. However the second rule violates both strong depth and strong weight-preservation.

- **Criteria by Extended Critical Pairs.** In [44], based on the preliminary work in [43], Jouannaud and Kirchner show that the union of a TRS $\mathcal{R}$ and a reversible TRS $\mathcal{S}$ is confluent if $\mathcal{R}/\mathcal{S}$ and $\rhd/{\leftrightarrow_{\mathcal{S}}}$ are terminating and

$$\mathcal{R}^{\leftarrow}\mathcal{S}^{\infty}\mathcal{R}^{\rightarrow} \cup \mathcal{R}\cup\mathcal{S}\cup\mathcal{S}^{-1} \subseteq {\to}^{*}_{\mathcal{R},\mathcal{S}} \cdot {\leftrightarrow}^{*}_{\mathcal{S}} \cdot {}^{*}_{\mathcal{R},\mathcal{S}}{\leftarrow}$$

  Here $s \to_{\mathcal{R},\mathcal{S}} t$ if there exist a rule $\ell \to r \in \mathcal{R}$, a position $p \in \mathcal{P}\mathrm{os}(s)$, and a substitution σ, such that $s|_p \leftrightarrow^*_{\mathcal{S}} \ell\sigma$ and $t = s[r\sigma]_p$. Note that $\mathcal{S}$ has a serious restriction: The two termination requirements prohibit application when $\mathcal{S}$ is erasing or collapsing, or even when $C[t] \leftrightarrow^*_{\mathcal{S}} t$ for a non-empty context C. For instance, Examples 6.1.8 and 6.1.9 cannot be handled due to this restriction. On the other hand it is applicable for mutually overlapping TRSs $\mathcal{R}$ and $\mathcal{S}$, for example:

$$1\colon\ x + x \to x \qquad 2\colon\ x + y \to y + x \qquad 3\colon\ (x + y) + z \to x + (y + z)$$

  By taking $\mathcal{R} = \{1\}$ and $\mathcal{S} = \{2,3\}$, one can easily show confluence of $\mathcal{R} \cup \mathcal{S}$ by using their criterion. However, Theorem 6.1.6 cannot be applied because $\mathcal{R}$ and $\mathcal{S}$ overlap on each other. This criterion forms a foundation of AC-completion.

- **Criteria by Relative Termination.** Geser [27] introduced several pioneering applications of relative termination. A result of particular interest in

this context is the following confluence criterion: A TRS $\mathcal{R} \cup \mathcal{S}$ is confluent if $\mathcal{R}$ is left-linear, $\mathcal{S}$ is confluent, and the inclusions

$$\mathcal{S}{\leftarrow}_{\varnothing}{}^{\infty}{\rightarrow}_{\mathcal{R}} \subseteq ({\rightarrow}_{\mathcal{S}}^{*} \cdot {}_{\mathcal{R}\cup\mathcal{S}}{}^{*}{\leftarrow}) \cup ({\rightarrow}_{\mathcal{R}} \cdot {\downarrow}_{\mathcal{R}\cup\mathcal{S}})$$

and

$$\mathcal{R}{\leftarrow}_{\varnothing}{}^{\infty}{\rightarrow}_{\mathcal{R}} \subseteq {\downarrow}_{\mathcal{R}\cup\mathcal{S}}$$

hold. In contrast to Theorem 6.1.6, overlaps between rules in $\mathcal{R}$ and $\mathcal{S}$ pose no problem. The following example, due to Geser, shows the power of his approach beyond pure left-linear systems:

$$1\colon\ \mathsf{c}(\mathsf{s}(x),\mathsf{s}(y)) \to \mathsf{c}(x,y) \qquad\qquad 2\colon\ \mathsf{c}(x,x) \to \mathsf{f}(\mathsf{c}(x,x))$$

Then confluence can be established by taking $\mathcal{R} = \{1\}$ and $\mathcal{S} = \{2\}$, whereas Theorem 6.1.6 is not applicable. [8] The reason for being able to handle overlaps between $\mathcal{R}$ and $\mathcal{S}$ is, that with the restriction of left-linearity of the $\mathcal{R}$-part, joinability of syntactical critical pairs suffices to establish confluence. On the other hand, the requirement of left-linearity prevents application for Examples 6.1.7, 6.1.8 and 6.1.9, except for choosing $\mathcal{R} = \varnothing$.

[8] All current confluence tools fail to show confluence of the one rule TRS consisting of rule number 2.

Chapter 7

Conclusion and Future Work

We have developed a new framework for completion, called maximal completion, that allows to characterize it as an optimization problem over constraints. The framework has been adapted to allow a similar maximality-based approach for various adaptations of completion to inductive theorem proving by means of constrained equalities. Moreover we showed a new and easily automatable criterion for confluence of non-left-linear and non-terminating TRSs.

In all the above our implementations followed very tightly the theoretical frameworks to show their practical effectiveness. Nevertheless, to make our results accessible and useful in other domains, highly practical tools have to be developed.

Maximal completion's main drawback is that it is currently restricted to methods of showing termination that can be encoded to Boolean constraints. It is therefore crucial to recapitulate termination techniques into maximal termination to extend the capability of `Maxcomp`. In particular, the dependency pair method [8] and semantic labelling [88] are known to be extremely powerful techniques of showing termination of TRSs. Techniques that to show maximal termination based on these methods would greatly benefit maximal completion.

The application of completion to theorem proving is an important direction. Several first order theorem provers like `Waldmeister` [53] are based on unfailing completion [12]. In Chapter 6 we briefly mention an extension of maximal completion. Due to its very ad-hoc nature these extensions are not covered in this thesis. A thorough investigation should be conducted on how to apply maximal completion in unfailing completion to construct more effective theorem provers.

Constrained Equalities give a framework to fully automate rewriting induction and other inductive proof methods. As mentioned in the introduction, the user provided reduction order is only one among many obstacles in constructing inductive proofs. Several methods have been proposed to extend rewriting induction, for example to deal with non-orientable equations [1] or to to construct needed lemmata automatically [71, 78, 3]. It is worthwhile to investigate if and how these techniques can be incorporated in our approach.

Last but not least, in Chapter 6 we presented a generalization of generalization

of Knuth and Bendix' confluence criterion, which can deal with non-left-linear, non-terminating TRSs and can be fully automated. As seen in Section 6.5, conditions required in our criterion are related to the results by Jouannaud and Kirchner [44] and Geser [27]. Any of them exploits relative termination to overcome the problem of non-termination, however still relative termination poses a strict restriction. In the case of left-linear TRSs, Hirokawa and Middeldorp employed *critical pair steps* [38] to relax this restriction. The resulting criterion generalizes both Knuth-Bendix' criterion and orthogonality (left-linearity and non-overlappingness). We anticipate that this and our result can be combined by using such critical pair steps. However deriving such a combined criterion is tightly related to the long-standing open conjecture that every strongly non-overlapping, right-linear TRS is confluent.[1]

[1]The RTA Open Problem #58: `http://rtaloop.mancoosi.univ-paris-diderot.fr/`

Appendix A

Experimental Data for Chapter 4

Table A.1 contains the list of all experimental results. Here $\times$ denotes failure, i.e. no more rules are orientable, no new critical pairs can be generated and the tool gives up. The symbol ∞ denotes that the tool in question did not provide a result within 600 seconds.

Table A.1: Experimental results for Maxcomp

	LPO		KBO		termination tool	
problem	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
AD93.Z22	∞	∞	∞	∞	∞	∞
ASK.93.1	0.01	0.10	0.02	0.01	0.01	0.98
ASK.93.2	∞	∞	∞	∞	∞	∞
ASK.93.5	∞	∞	∞	∞	∞	∞
ASK.93.6	16.75	1.06	10.93	0.83	5.73	10.81
aufgabe3.2	0.01	0.00	0.01	0.01	0.01	0.95
aufgabe3.3	0.01	0.01	0.01	0.01	0.01	0.87
BD94.collapse	0.05	0.01	0.08	0.01	0.03	1.18
BD94.peano	0.03	0.01	∞	∞	0.02	1.22
BD94.sqrt	0.02	0.00	0.03	0.01	0.02	1.00
BGK94.D08	∞	0.70	∞	∞	∞	∞
BGK94.D10	∞	0.88	∞	∞	51.49	∞
BGK94.D12	∞	1.18	∞	∞	52.44	∞
BGK94.D16	∞	37.48	∞	∞	74.91	∞
BGK94.M08	0.74	0.24	1.25	0.49	0.59	7.72
BGK94.M10	1.20	0.81	2.24	∞	0.94	20.97
BGK94.M12	1.83	41.69	3.65	∞	1.42	27.18
BGK94.M14	2.73	∞	5.73	∞	2.06	25.88
BGK94.Z22W	∞	∞	∞	∞	∞	∞

continued on next page

continued from previous page

	LPO		KBO		termination tool	
problem	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
BH96.fac8	0.11	0.02	0.52	0.06	0.07	2.07
Chr89.A2	4.21	0.99	4.81	0.95	95.76	182.93
Chr89.A24	∞	∞	∞	∞	∞	∞
Chr89.A3	40.06	3.17	20.85	∞	∞	∞
fggx	0.01	0.01	0.02	0.01	0.01	0.98
fib	0.87	100.32	∞	∞	0.85	7.18
HR94.1	∞	∞	∞	∞	∞	∞
HR94.2	∞	∞	∞	∞	∞	∞
kb.fail	×	×	×	×	×	×
kb.fail1	×	×	×	×	×	×
KK99.linear_assoc	0.02	0.01	0.02	0.01	0.03	0.83
Les83.fib	0.10	0.01	∞	∞	0.23	2.82
Les83.subset	0.22	0.01	∞	∞	0.09	3.27
lr.theory	0.35	0.11	0.45	2.89	1.74	7.90
LS06.CGE4	∞	∞	∞	∞	∞	∞
LS06.CGE5	∞	∞	∞	∞	∞	∞
LS94.G0	0.52	0.05	0.71	0.06	1.28	5.90
LS94.G1	∞	∞	∞	∞	∞	∞
LS94.G2	∞	∞	∞	∞	∞	∞
LS94.G3	∞	∞	∞	∞	∞	∞
LS94.P1	∞	∞	∞	∞	∞	∞
OKW95.dt1	0.78	46.15	∞	∞	0.91	6.22
rl.theory	2.36	0.17	∞	1.21	1.76	8.07
Sim91_sims2	∞	∞	∞	∞	∞	∞
SK90.3.01	1.20	0.09	2.23	0.61	4.16	18.85
SK90.3.02	0.03	0.01	0.03	0.02	0.05	0.97
SK90.3.03	4.69	1.01	3.88	0.05	1.59	5.54
SK90.3.04	6.72	1.75	∞	∞	99.98	∞
SK90.3.05	1.58	0.35	1.07	0.33	3.46	∞
SK90.3.06	4.22	0.90	∞	∞	∞	∞
SK90.3.07	∞	3.03	∞	∞	∞	∞
SK90.3.08	0.08	0.01	0.06	0.03	0.05	1.18
SK90.3.09	∞	∞	∞	∞	∞	∞
SK90.3.10	0.03	0.01	0.03	0.01	0.02	0.92
SK90.3.11	0.03	0.01	0.03	0.01	0.02	1.02
SK90.3.12	×	×	×	×	×	×
SK90.3.13	0.17	0.02	0.16	0.02	0.17	1.33
SK90.3.14	0.13	0.01	0.21	0.16	0.15	1.65
SK90.3.15	∞	∞	0.12	0.02	0.08	1.39

continued on next page

continued from previous page

	LPO		KBO		termination tool	
problem	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
SK90.3.16	0.02	0.01	0.03	0.01	0.01	0.80
SK90.3.17	0.05	0.01	0.04	0.01	0.08	1.25
SK90.3.18	0.15	0.01	∞	∞	0.30	3.50
SK90.3.19	0.15	0.09	0.26	15.11	0.32	2.73
SK90.3.20	0.35	1.34	0.42	∞	0.46	2.65
SK90.3.21	0.16	0.04	0.25	0.03	0.19	85.81
SK90.3.22	∞	4.03	∞	8.03	∞	∞
SK90.3.23	0.03	96.93	0.43	0.19	0.37	4.72
SK90.3.24	0.04	0.02	0.04	0.01	0.04	1.26
SK90.3.25	0.02	0.43	0.02	0.01	0.03	0.99
SK90.3.26	∞	∞	∞	∞	∞	∞
SK90.3.27	13.31	21.09	36.47	0.53	72.31	∞
SK90.3.28	∞	20.23	53.58	∞	∞	110.76
SK90.3.29	2.29	0.86	3.91	1.75	2.02	5.27
SK90.3.30	0.03	0.01	0.03	0.02	0.02	0.76
SK90.3.31	0.03	0.01	0.04	0.01	0.01	0.84
SK90.3.32	0.01	0.01	0.01	0.01	0.01	0.74
SK90.3.33	0.03	0.01	0.02	0.01	0.02	0.91
sl.ackermann	0.02	0.01	∞	∞	0.03	0.68
sl.cge	∞	∞	∞	∞	173.03	∞
sl.cge3	∞	∞	∞	∞	∞	∞
sl.endo	∞	0.62	∞	0.38	3.88	7.60
sl.equiv.proofs	×	×	∞	∞	2.52	262.32
sl.equiv.proofs.or	×	×	∞	∞	2.81	∞
sl.fgh	0.01	0.01	0.01	0.01	0.01	0.73
sl.groups	0.45	0.08	∞	0.08	0.54	2.22
sl.groups_conj	0.11	0.02	0.18	1.10	0.21	2.49
sl.hard	0.01	0.01	0.01	0.02	0.01	0.69
slothrop.nlp-2b	∞	∞	∞	∞	∞	∞
TPDB.torpa.secr10	2.21	0.17	14.34	0.55	9.15	50.46
TPDB.torpa.secr4	4.77	35.69	15.25	31.62	17.10	35.56
TPDB.thiemann27	∞	∞	∞	∞	∞	∞
TPDB.zantema.z115	∞	0.73	5.76	42.63	15.27	203.69
TPTP.BOO027-1	0.04	0.02	0.04	0.01	0.03	1.37
TPTP.COL053-1	0.01	0.01	0.01	0.01	0.01	0.67
TPTP.COL056-1	0.04	0.01	0.03	0.87	0.03	0.82
TPTP.COL060-1	×	×	0.01	0.01	0.02	1.14
TPTP.COL085-1	0.01	0.01	0.01	0.01	0.01	0.66
TPTP.GRP010-4	1.52	0.27	2.21	0.11	6.02	231.47

continued on next page

continued from previous page

	LPO		KBO		termination tool	
problem	mkbTT	Maxcomp	mkbTT	Maxcomp	mkbTT	Slothrop
TPTP.GRP011-4	0.48	0.09	0.62	0.10	0.63	25.42
TPTP.GRP012-4	0.28	0.03	0.34	0.03	0.48	2.13
TPTP.GRP393-2	0.01	0.01	0.01	0.01	0.01	0.68
TPTP.GRP394-3	0.33	0.06	0.42	0.07	0.51	2.68
TPTP.GRP454-1	17.13	2.65	49.63	0.14	109.87	∞
TPTP.GRP457-1	17.18	2.66	49.98	1.62	112.73	∞
TPTP.GRP460-1	∞	0.89	17.74	2.91	16.99	∞
TPTP.GRP463-1	∞	1.76	17.80	3.26	16.91	∞
TPTP.GRP481-1	∞	2.40	∞	57.20	∞	69.56
TPTP.GRP484-1	∞	0.52	∞	13.52	∞	∞
TPTP.GRP487-1	∞	1.00	∞	7.05	∞	×
TPTP.GRP490-1	∞	0.84	168.93	13.21	∞	∞
TPTP.GRP493-1	∞	5.13	40.54	2.83	∞	∞
TPTP.GRP496-1	∞	0.90	∞	11.13	∞	241.10
TPTP.HWC004-1	0.13	0.03	0.09	0.01	0.14	1.32
TPTP.HWC004-2	0.05	0.01	0.04	0.01	0.03	1.09
TPTP.SWV262-2	0.03	0.02	0.03	0.05	0.03	0.88
WS06.proofreduction	∞	∞	∞	∞	162.66	∞

Appendix B

Experimental Data for Chapter 5

Table B.2 contains the list of all experimental results of completion based on constrained equalities, and Table B.1 contains all data for rewriting induction based on constrained equalities. Here $\times$ denotes failure, i.e. that no more rules are orientable, no new critical pairs or expansions can be generated and the tool gives up. For rewriting induction with constrained equalities we do not provide precise timing information, instead Y denotes a successful proof. For completion the symbol ∞ denotes a timeout. The time limit was set to 60 seconds for completion and to 10 seconds for the inductive proofs.

Table B.1: Experimental results for Induction with Constraint Equalities

problem	LPO	KBO	*problem*	LPO	KBO
DC #2	Y	Y	DC #3	∞	∞
DC #4	∞	∞	DC #5	Y	Y
DC #7	Y	Y	DC #22	Y	Y
DC #23	Y	Y	DC #25	∞	∞
DC #27	∞	∞	DC #28	∞	∞
DC #29	∞	∞	DC #30	∞	∞
DC #31	∞	∞	DC #43	∞	∞
DC #47	∞	∞	DC #55	∞	Y
DC #58	Y	∞	DC #60	∞	∞
DC #63	∞	∞	DC #64	∞	∞
DC #79	∞	∞	DC #80	∞	∞
DC #81	∞	∞	DC #82	∞	∞
DC #83	∞	∞	DC #84	∞	∞
DC #109	∞	∞	DC #110	Y	Y
DC #111	∞	∞	DC #112	Y	Y
DC #113	∞	∞	DC #115	Y	Y

continued on next page

continued from previous page

problem	LPO	KBO	*problem*	LPO	KBO
DC #116	∞	∞	DC #158	∞	∞
DC #208	Y	∞	DC #216	∞	∞
DC #225	Y	Y	DC #228	Y	Y
DC #229	Y	Y	DC #230	Y	Y
DC #231	Y	∞	DC #232	∞	∞
DC #233	∞	∞	DC #234	Y	∞
DC #236	∞	∞	DC #237	∞	∞
DC #270	∞	∞	DC #281	Y	Y
DC #283	Y	Y	DC #287	Y	Y
DC #300	∞	∞	DC #301	Y	Y
DC #305	Y	∞	DC #318	∞	∞
DC #319	∞	∞	DC #349	Y	∞
DC #370	∞	∞	DC #375	∞	∞
DC #442	Y	∞	DC #502	Y	∞
DC #665	Y	Y	DC #694	∞	∞
DC #699	Y	∞	DC #704	Y	Y
DC #744	Y	Y	DC #1003	Y	Y
DC #1018	Y	Y	DC #1052	∞	∞
DC #1107	∞	Y	mul	Y	∞
mul2	∞	∞	sato1	Y	∞
sato2	∞	∞			

Table B.2: Experimental results for Completion with Constraint Equalities

problem	LPO	KBO	*problem*	LPO	KBO
AD93.Z22	∞	∞	ASK.93.1	0.26	∞
ASK.93.2	∞	∞	ASK.93.5	∞	∞
ASK.93.6	∞	∞	aufgabe3.2	0.01	0.01
aufgabe3.3	0.01	0.01	BD94.collapse	0.05	0.02
BD94.peano	0.01	∞	BD94.sqrt	0.01	0.01
BGK94.D08	∞	∞	BGK94.D10	∞	∞
BGK94.D12	∞	∞	BGK94.D16	∞	∞
BGK94.M08	∞	∞	BGK94.M10	∞	∞
BGK94.M12	∞	∞	BGK94.M14	∞	∞
BGK94.Z22W	∞	∞	BH96.fac8	0.03	0.03
Chr89.A2	∞	2.38	Chr89.A24	∞	∞

continued on next page

continued from previous page

problem	LPO	KBO	*problem*	LPO	KBO
Chr89.A3	∞	∞	fggx	0.01	0.02
fib	∞	∞	HR94.1	∞	∞
HR94.2	∞	∞	kb.fail	×	×
kb.fail1	×	×	KK99.linear_assoc	0.01	0.01
Les83.fib	0.03	∞	Les83.subset	0.01	∞
lr.theory	10.04	∞	LS06.CGE4	∞	∞
LS06.CGE5	∞	∞	LS94.G0	∞	∞
LS94.G1	∞	∞	LS94.G2	∞	∞
LS94.G3	∞	∞	LS94.P1	∞	∞
OKW95.dt1	∞	∞	rl.theory	1.22	∞
Sim91_sims2	∞	∞	SK90.3.01	1.03	∞
SK90.3.02	0.01	∞	SK90.3.03	4.69	∞
SK90.3.04	∞	∞	SK90.3.05	∞	3.13
SK90.3.06	∞	∞	SK90.3.07	∞	∞
SK90.3.08	0.03	0.07	SK90.3.09	∞	∞
SK90.3.10	0.01	0.01	SK90.3.11	0.07	0.08
SK90.3.12	×	×	SK90.3.13	0.05	0.03
SK90.3.14	0.11	∞	SK90.3.15	∞	0.06
SK90.3.16	0.01	0.01	SK90.3.17	0.01	∞
SK90.3.18	∞	0.01	SK90.3.19	0.27	∞
SK90.3.20	0.64	∞	SK90.3.21	4.50	0.08
SK90.3.22	∞	4.03	SK90.3.23	35.67	∞
SK90.3.24	0.06	0.04	SK90.3.25	0.18	0.01
SK90.3.26	∞	∞	SK90.3.27	∞	∞
SK90.3.28	∞	30.88	SK90.3.29	0.03	∞
SK90.3.30	0.03	0.03	SK90.3.31	0.02	0.01
SK90.3.32	0.01	0.01	SK90.3.33	0.01	0.01
sl.ackermann	0.01	∞	sl.cge	∞	∞
sl.cge3	∞	∞	sl.endo	∞	∞
sl.equiv.proofs	×	×	sl.equiv.proofs.or	×	×
sl.fgh	0.01	0.03	sl.groups	0.46	∞
sl.groups_conj	0.08	0.08	sl.hard	0.01	∞
slothrop.nlp-2b	∞	∞	TPDB.torpa.secr10	∞	17.39
TPDB.torpa.secr4	∞	∞	TPDB.thiemann27	∞	∞
TPDB.zantema.z115	∞	∞	TPTP.BOO027-1	0.01	0.02
TPTP.COL053-1	0.01	0.01	TPTP.COL056-1	0.01	0.01
TPTP.COL060-1	×	0.01	TPTP.COL085-1	0.01	0.01
TPTP.GRP010-4	∞	∞	TPTP.GRP011-4	1.63	∞
TPTP.GRP012-4	0.10	0.10	TPTP.GRP393-2	0.01	0.01

continued on next page

continued from previous page

problem	LPO	KBO	*problem*	LPO	KBO
TPTP.GRP394-3	0.29	∞	TPTP.GRP454-1	∞	∞
TPTP.GRP457-1	∞	∞	TPTP.GRP460-1	∞	∞
TPTP.GRP463-1	∞	∞	TPTP.GRP481-1	∞	∞
TPTP.GRP484-1	∞	∞	TPTP.GRP487-1	∞	∞
TPTP.GRP490-1	∞	∞	TPTP.GRP493-1	∞	∞
TPTP.GRP496-1	∞	∞	TPTP.HWC004-1	0.01	0.01
TPTP.HWC004-2	0.01	0.01	TPTP.SWV262-2	0.03	∞
WS06.proofreduction	∞	∞			

Appendix C

Experimental Data for Chapter 6

Table C.1: Experimental results for the new confluence criterion

problem	ACP	ACP*	CSI	CSI*	Saigawa	Saigawa*
Example 6.1.7	?(0.07)	Y(0.49)	?(6.32)	Y(13.62)	?(0.19)	Y(0.55)
Example 6.1.8	?(0.07)	Y(0.43)	?(1.81)	Y(10.77)	?(0.19)	Y(0.61)
Example 6.1.9	?(0.07)	Y(1.03)	?(6.32)	Y(20.19)	?(0.25)	Y(1.15)
Cops #1	Y(0.19)	Y(0.67)	Y(0.67)	Y(1.11)	?(0.31)	?(2.29)
Cops #2	Y(0.07)	Y(0.07)	Y(0.67)	Y(2.04)	?(0.19)	?(5.55)
Cops #3	Y(0.19)	Y(0.13)	Y(0.79)	Y(2.05)	?(0.25)	?(4.77)
Cops #15	N(0.43)	N(0.43)	∞	∞	?(0.19)	?(0.19)
Cops #16	?(0.13)	?(2.78)	?(2.96)	?(12.40)	?(0.19)	?(2.89)
Cops #17	Y(0.19)	Y(0.07)	?(6.39)	?(39.52)	?(0.19)	?(13.72)
Cops #24	?(0.07)	?(3.44)	?(6.33)	?(14.94)	?(0.19)	?(3.56)
Cops #25	Y(0.07)	Y(0.67)	?(0.67)	Y(1.11)	?(0.19)	Y(0.85)
Cops #26	?(0.07)	?(3.44)	?(6.29)	?(14.82)	?(0.19)	?(3.57)
Cops #27	?(0.07)	?(1.71)	?(2.66)	?(25.45)	?(0.19)	?(2.24)
Cops #28	?(0.07)	Y(0.79)	?(6.33)	Y(11.96)	?(0.19)	Y(0.91)
Cops #46	N(0.07)	N(0.07)	N(0.37)	N(1.33)	?(0.19)	N(0.61)
Cops #47	?(0.07)	?(3.44)	?(6.33)	?(16.17)	?(0.19)	Y(3.62)
Cops #53	Y(0.25)	Y(0.13)	Y(0.43)	Y(1.72)	?(0.31)	?(6.51)
Cops #55	Y(0.31)	Y(0.13)	Y(0.43)	Y(1.74)	?(0.31)	?(5.43)
Cops #59	?(0.13)	Y(0.49)	?(1.82)	Y(7.75)	?(0.19)	Y(0.61)
Cops #70	?(0.13)	N(0.85)	?(0.31)	N(1.24)	?(0.19)	N(0.97)
Cops #76	?(0.13)	?(0.73)	∞	∞	?(0.19)	?(0.91)
Cops #77	?(0.37)	?(13.32)	∞	∞	?(0.19)	?(13.43)
Cops #78	?(0.37)	?(2.90)	∞	∞	?(0.19)	?(2.84)
Cops #79	Y(0.07)	Y(0.07)	?(6.32)	?(21.41)	?(0.19)	?(3.62)
Cops #89	Y(0.07)	Y(0.07)	?(6.35)	?(35.35)	?(0.19)	?(2.24)

continued on next page

continued from previous page

problem	ACP	ACP*	CSI	CSI*	Saigawa	Saigawa*
Cops #91	?(0.07)	Y(1.16)	?(6.32)	Y(20.05)	?(0.19)	Y(1.28)
Cops #94	Y(0.13)	Y(0.13)	Y(4.52)	Y(7.01)	?(0.19)	Y(31.82)
Cops #97	Y(0.07)	Y(0.07)	Y(0.67)	Y(4.05)	?(0.19)	Y(0.73)
Cops #98	N(0.07)	N(0.07)	N(0.37)	N(3.38)	?(0.19)	?(1.16)
Cops #107	?(0.07)	Y(0.49)	?(1.95)	Y(2.42)	?(0.19)	Y(0.61)
Cops #108	Y(0.13)	Y(0.07)	?(6.33)	?(11.08)	?(0.49)	?(5.01)
Cops #109	?(0.43)	?(2.96)	∞	∞	?(0.19)	?(2.77)

Appendix D

Maxcomp: Installation and Usage

Maxcomp is a fully automatic completion tool for equational systems for Linux based operating systems, written in the OCaml programming language. In this section we briefly describe its set-up and usage.

Installation

The source code of Maxcomp is available on-line at

`http://www.jaist.ac.jp/project/maxcomp`

Maxcomp builds heavily on the works of others. Unfortunately, this makes compilation and installation slightly tedious since several software packages are needed to compile the source code. Some (OCaml, GNU make, Findlib, CamlIDL, GMP) might be available through the package system of your distribution. Below, version numbers refer to those versions of the packages which were used during the development of Maxcomp. Later versions might work, but backward-compatibility is not guaranteed. The packages are:

- Objective CAML (OCaml) 3.12, available at

  `http://www.caml.inria.fr`

- The GNU make tool 3.18, available at

  `http://www.gnu.org/s/make`

- OCaml Findlib 1.2.7, available at

  `http://projects.camlcity.org/projects/findlib.html`

- Ocamlyices 7bd6030, available at

  `https://github.com/polazarus/ocamlyices`

OCaml Findlib should be installed before installing Ocamlyices. Moreover after installing Findlib, the following packages, which are needed for Ocamlyices, have to be installed. They should be installed in the following order:

1. CamlIDL 1.05.13, available at

   `http://forge.ocamlcore.org/projects/camlidl`

2. The GNU Multiple Precision Arithmetic Library (GMP) 2.5.1, available at

   `http://gmplib.org`

   This is only needed if you do not install Yices with statically linked GMP libraries. It is recommended however to install Yices with non-statically linked GMP libraries.

3. Yices 1.0.29, available at

   `http://yices.csl.sri.com`

After installation of all these packages and Ocamlyices, extract the contents of the archive of `Maxcomp` into a folder:

```
tar xzvf maxcomp-1.0.1.tar.gz
```

Switch to the newly generated directory, and execute make

```
cd maxcomp-1.0.1
make
```

to compile and generate the `Maxcomp` binary.

Usage

The general usage of `Maxcomp` is

```
maxcomp <options> <file>
```

If successful, it will output a line `YES` followed by the found complete TRS. Available input options are:

- -lpo (default)
  this option encodes constraints based on *lexicographic path orders (LPO)* to find maximal terminating TRSs

- -mpo
  this option encodes constraints based on *multi-set path orders (MPO)* to find maximal terminating TRSs

- -kbo
  this option encodes constraints based on *Knuth-Bendix orders (KBO)* to find maximal terminating TRSs

- -mi <d,n>
  this option encodes constraints based on $d * d$ *matrix interpretations* with coefficients $\leq n$

- -K <n>
  this option enforces to initially compute $n - 1$ maximal terminating TRSs. If omitted, the default value is set to 2, i.e. only one maximal terminating TRS is computed

Only one of the options -lpo, -mpo, -kbo and -mi <d,n> can be activated at a time. If multiple of those options are specified, all but the last one are ignored. We give some examples:

```
maxcomp filename.es
```

tries to complete the equational system specified in filename.es by encoding constraints based on lexicographic path orders with parameter K initially set to 2.

```
maxcomp -kbo -mi 2 2 -K 4 filename.es
```

tries to complete the equational system specified in filename.es by encoding constraints based on $2 * 2$ matrix interpretations with coefficients ≤ 2, and parameter K initially set to 4. Note that the option -kbo is ignored and could be simply omitted.

As input, Maxcomp accepts the TPDB[1] plain text file format. To give some intuition, consider the specification of group-theory, sl_groups.trs:

```
(VAR x y z )
(RULES
f(x,f(y,z)) ->  f(f(x,y),z)
f(x,i(x)) -> e
f(x,e) ->  x
)
```

Roughly, the first line lists all variables in the subsequent rules. The TPDB format was intended to to specify term rewriting systems, and not equational systems, but due to its popularity, it is used as the input format of Maxcomp. Intended for TRSs, a direction -> is indicated between terms. Maxcomp however ignores this direction and treats the specified rules as equations. Therefore writing e.g.

```
e -> f(x,i(x))
```

instead of `f(x,i(x)) -> e` is still valid input and specifies the same equational system, even though it violates the variable condition of TRSs when interpreted as a rule.

[1] http://www.lri.fr/~marche/tpdb/format.html

References

[1] Takahito Aoto. Dealing with non-orientable equations in rewriting induction. In *Proc. 17th International Conference on Rewriting Techniques and Applications*, volume 4098 of *Lecture Notes in Computer Science*, pages 242–256, 2006.

[2] Takahito Aoto. Designing a rewriting induction prover with an increased capability of non-orientable theorems. In *Proc. Austrian-Japanese Workshop on Symbolic Computation in Software Science*, volume 08-08 of *RISC Technical Report*, pages 1–15, 2008.

[3] Takahito Aoto. Sound lemma generation for proving inductive validity of equations. In *Proc. IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 2 of *Leipzig International Proceedings in Computer Science*, pages 13–24, 2008.

[4] Takahito Aoto, Nao Hirokawa, Dominik Klein, and Sarah Winkler. Constrained equalities. Under preparation. 2012.

[5] Takahito Aoto and Yoshihito Toyama. Persistency of confluence. *Journal of Universal Computer Science*, 3(11):1134–147, 1997.

[6] Takahito Aoto and Yoshihito Toyama. A reduction-preserving completion for proving confluence of non-terminating term rewriting systems. *Logical Methods in Computer Science*, 8(1-31):1–29, 2012.

[7] Takahito Aoto, Junichi Yoshida, and Yoshihito Toyama. Proving confluence of term rewriting systems automatically. In *Proc. 21st International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 93–102, 2009.

[8] Thomas Arts and Jürgen Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[9] Jürgen Avenhaus. *Reduktionssysteme. Rechnen und Schließen in gleichungsdefinierten Strukturen*. Springer, 1995.

[10] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[11] Leo Bachmair and Nachum Dershowitz. Equational inference, canonical proofs, and proof orderings. *Journal of the ACM*, 41(2):236–276, 1994.

[12] Leo Bachmair, Nachum Dershowitz, and Jieh Hsiang. Orderings for equational proofs. In *Proc. ACM/IEEE Symposium on Logic in Computer Science*, pages 346–357, 1986.

[13] Armin Biere, Alessandro Cimatti, Edmund M. Clarke, and Yunshan Zhu. Symbolic model checking without BDDs. In *Proc. 5th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, volume 1579 of *Lecture Notes in Computer Science*, pages 193–207, 1999.

[14] Garrett Birkhoff. On the structure of abstract algebras. *Proceedings of the Cambridge Philosophical Society*, 31(4):433–454, 1935.

[15] Alan Bundy. The automation of proof by mathematical induction. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 845–911. Elsevier and MIT Press, 2001.

[16] Alonzo Church and John B. Rosser. Some properties of conversion. *Transaction of the American Mathematical Society*, 39:472–482, 1936.

[17] Michael Codish, Vitaly Lagoon, and Peter J. Stuckey. Solving partial order constraints for LPO termination. *JSAT*, 5(1–4):193–215, 2008.

[18] Hubert Comon. Inductionless induction. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, pages 913–962. Elsevier and MIT Press, 2001.

[19] Martin Davis, George Logemann, and Donald W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[20] Nachum Dershowitz. A note on simplification orderings. *Information Processing Letters*, 9(5):212–215, 1979.

[21] Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.

[22] Bruno Dutertre and Leonardo De Moura. A fast linear-arithmetic solver for DPLL(T). In *Proc. 18th International Conference on Computer Aided Verification*, volume 4144 of *Lecture Notes in Computer Science*, pages 81–94, 2006.

[23] Jörg Endrullis, Johannes Waldmann, and Hans Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40(2-3):195–220, 2008.

[24] Michael J. Fay. First-order unification in an equational theory. In *Proc. 4th Workshop on Automated Deduction*, pages 161–167, 1979.

[25] Bertram Felgenhauer, Harald Zankl, and Aart Middeldorp. Layer systems for proving confluence. In *Proc. 31st International Conference on Foundations of Software Technology and Theoretical Computer Science*, volume 13 of *Leipzig International Proceedings in Computer Science*, pages 288–299, 2011.

[26] Carsten Fuhs, Jürgen Giesl, Aart Middeldorp, Peter Schneider-Kamp, René Thiemann, and Harald Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proc. of the 10th International Conference on Theory and Applications of Satisfiability Testing*, volume 4501 of *Lecture Notes in Computer Science*, pages 340–354, 2007.

[27] Alfons Geser. *Relative Termination*. PhD thesis, Universität Passau, 1990. Available as technical report 91-03.

[28] Jürgen Giesl, Peter Schneider-Kamp, and René Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proc. 3rd International Joint Conference on Automated Reasoning*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 281–286, 2006.

[29] Jürgen Giesl, René Thiemann, and Peter Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proc. 5th International Symposium on Frontiers of Combining Systems*, volume 3717 of *Lecture Notes in Artificial Intelligence*, pages 216–231, 2005.

[30] Jürgen Giesl, René Thiemann, Peter Schneider-Kamp, and Stephan Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[31] Hiroshi Gomi, Michio Oyamaguchi, and Yoshikatsu Ohta. On the Church-Rosser property of non-E-overlapping and strongly depth-preserving term rewriting systems. *Trans. IPSJ*, 37(12):2147–2160, 1996.

[32] Hiroshi Gomi, Michio Oyamaguchi, and Yoshikatsu Ohta. On the Church-Rosser property of non-E-overlapping and weight-preserving TRS's. Technical Report 950, RIMS Research Report, 1996.

[33] Hiroshi Gomi, Michio Oyamaguchi, and Yoshikatsu Ohta. On the Church-Rosser property of root-E-overlapping and strongly depth-preserving term rewriting systems. *Trans. IPSJ*, 39(4):992–1005, 1998.

[34] Bernhard Gramlich. Completion based inductive theorem proving: An abstract framework and its applications. In *Proc. 9th European Conf. on Artificial Intelligence*, pages 314–319. Pitman Publishing, London, 1990.

[35] Bernhard Gramlich. Confluence without termination via parallel critical pairs. In *Proc. 21st International Colloquium, Linköping Trees in Algebra and Programming*, volume 1059 of *Lecture Notes in Computer Science*, pages 211–225, 1996.

[36] Bernhard Gramlich. Strategic issues, problems and challenges in inductive theorem proving. *Electronic Notes in Theoretical Computer Science*, 125(2):5–43, 2005.

[37] Nao Hirokawa and Aart Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1-2):172–199, 2005.

[38] Nao Hirokawa and Aart Middeldorp. Decreasing diagrams and relative termination. *Journal of Automated Reasoning*, 47:481–501, 2011.

[39] Gerald Huet. Confluent reductions: Abstract properties and applications to term rewriting systems. *Journal of the ACM*, 27:797–821, 1980.

[40] Gerald Huet. A complete proof of correctness of the Knuth-Bendix completion algorithm. *Journal of Computer and System Sciences*, 21(1):11–21, 1981.

[41] Gérard Huet and Dallas Lankford. On the uniform halting problem for term rewriting systems. Technical Report 283, INRIA, 1981.

[42] Jean-Marie Hullot. Canonical forms and unification. In *Proc. 5th International Conference on Automated Deduction*, volume 87 of *Lecture Notes in Computer Science*, pages 318–334, 1980.

[43] Jean-Pierre Jouannaud. Confluent and coherent equational term rewriting systems: Application to proofs in abstract data types. In *Proc. 8th International Colloquium on Trees in Algebra and Programming*, volume 159 of *Lecture Notes in Computer Science*, pages 269–283, 1983.

[44] Jean-Pierre Jouannaud and Hélène Kirchner. Completion of a set of rules modulo a set of equations. *SIAM Journal on Computing*, 15(4):1155–1194, 1986.

[45] Sam Kamin and Jean J. Lévy. Two generalizations of the recursive path ordering. Unpublished manuscript, University of Illinois, 1980.

[46] Dominik Klein and Nao Hirokawa. Maximal completion. In *Proc. 22nd International Conference on Rewriting Techniques and Applications*, volume 10 of *Leipzig International Proceedings in Computer Science*, pages 71–80, 2011.

[47] Dominik Klein and Nao Hirokawa. Confluence of non-left-linear TRSs via relative termination. In *Proc. 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 7180 of *Lecture Notes in Computer Science*, pages 258–273, 2012.

[48] Jan W. Klop. *Combinatory reduction systems*. PhD thesis, Utrecht University, 1980.

[49] Donald E. Knuth and Peter Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, 1970.

[50] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean Termination Tool 2. In *Proc. 20th International Conference on Rewriting Techniques and Applications*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304, 2009.

[51] Masahito Kurihara and Hisashi Kondo. Completion for multiple reduction orderings. *Journal of Automated Reasoning*, 23(1):25–42, 1999.

[52] Masahito Kurihara and Hisashi Kondo. Efficient BDD encodings for partial order constraints with application to expert systems in software verification. In *Proc. 17th International Conference on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems*, volume 3029 of *Lecture Notes in Artificial Intelligence*, pages 827–837, 2004.

[53] Bernd Löchner and Thomas Hillenbrand. A phytography of WALDMEISTER. *AI Communications*, 15(2–3):127–133, 2002.

[54] Yves Métivier. About the rewriting systems produced by the Knuth-Bendix completion algorithm. *Information Processing Letters*, 16(1):31–34, 1983.

[55] Robin Milner, Mads Tofte, Robert Harper, and David MacQueen. *The Definition of Standard ML, Revised Edition*. MIT Press, 1997.

[56] Maxwell H.A. Newman. On theories with a combinatorial definition of "equivalence". *Annals of Mathematics*, 43(2):223–243, 1942.

[57] Mizuhito Ogawa. Chew's theorem revisited - uniquely normalizing property of nonlinear term rewriting systems-. In *Proc. 3rd International Symposium on Algorithms and Computation*, volume 650 of *Lecture Notes in Computer Science*, pages 309–318, 1992.

[58] Enno Ohlebusch. *Modular Properties of Composable Term Rewriting Systems*. PhD thesis, Universität Bielefeld, 1994.

[59] Enno Ohlebusch. Modular properties of composable term rewriting systems. *Journal of Symbolic Computation*, 20:1–41, 1995.

[60] Enno Ohlebusch. *Advanced topics in term rewriting*. Springer, 2002.

[61] Simon Peyton Jones et al. The Haskell 98 language and libraries: The revised report. *Journal of Functional Programming*, 13(1):0–255, Jan 2003.

[62] Uday S. Reddy. Term rewriting induction. In *Proc. 10th International Conference on Automated Deduction*, volume 449 of *Lecture Notes in Computer Science*, pages 162–177, 1990.

[63] John Alan Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[64] Barry K. Rosen. Tree-manipulating systems and Church-Rosser theorems. *Journal of the ACM*, 20(1):160–187, 1973.

[65] Masahiko Sakai and Mizuhito Ogawa. Weakly-non-overlapping non-collapsing shallow term rewriting systems are confluent. *Information Processing Letters*, 110(18–19):810–814, 2010.

[66] Haruhiko Sato and Masahito Kurihara. Multi-context rewriting induction with termination checkers. *IEICE Transactions*, 93-D(5):942–952, 2010.

[67] Haruhiko Sato, Masahito Kurihara, Sarah Winkler, and Aart Middeldorp. Constraint-based multi-completion procedures for term rewriting systems. *IEICE Transactions*, 92-D(2):220–234, 2009.

[68] Haruhiko Sato, Sarah Winkler, Masahito Kurihara, and Aart Middeldorp. Multi-completion with termination tools (system description). In *Proc. 4th International Joint Conference on Automated Reasoning*, Lecture Notes in Computer Science, pages 306–312, 2008.

[69] Andrea Sattler-Klein. About changing the ordering during Knuth-Bendix completion. In *Proc. 11th Annual Symposium on Theoretical Aspects of Computer Science*, volume 775 of *Lecture Notes in Computer Science*, pages 175–186, 1994.

[70] Peter Schneider-Kamp, René Thiemann, Elena Annov, Michael Codish, and Jürgen Giesl. Proving termination using recursive path orders and SAT solving. In *Proc. of the 6th International Symposium Frontiers of Combining Systems*, volume 4720 of *Lecture Notes in Computer Science*, pages 267–282, 2007.

[71] Satoshi Shimazu, Takahito Aoto, and Yoshihito Toyama. Automated lemma generation for rewriting induction with disproof. *Computer Software*, 26(2):41–55, 2009. (in Japanese).

[72] Joachim Steinbach and Ulrich Kühler. Check your ordering—termination proofs and open problems. Technical Report SR-90-25, Universität Kaiserslautern, 1990.

[73] Aaron Stump, Garrin Kimmell, and Roba El Haj Omar. Type preservation as a confluence problem. In *Proc. 22nd International Conference on Rewriting Techniques and Applications*, volume 10 of *Leipzig International Proceedings in Computer Science*, pages 345–360, 2011.

[74] Aaron Stump and Bernd Löchner. Knuth-Bendix completion of theories of commuting group endomorphisms. *Information Processing Letters*, 98(5):195–198, 2006.

[75] TeReSe. *Term Rewriting Systems*, volume 55 of *Cambridge Tracts in Theoretical Computer Science*. Cambridge University Press, 2003.

[76] Yoshihito Toyama. On the Church-Rosser property for the direct sum of term rewriting systems. *Journal of the ACM*, 34(1):128–143, 1987.

[77] Yoshihito Toyama. Commutativity of term rewriting systems. In K. Fuchi and L. Kott, editors, *Programming of Future Generation Computers II*, pages 393–407. North-Holland, 1988.

[78] Pascal Urso and Emmanuel Kounalis. Sound generalizations in mathematical induction. *Theoretical Computer Science*, 323(1-3):443–471, 2004.

[79] Vincent van Oostrom. Confluence by decreasing diagrams. *Theoretical Computer Science*, 126(2):259–280, 1994.

[80] Vincent van Oostrom. Development closed critical pairs. In *Higher-Order Algebra, Logic, and Term Rewriting, Second International Workshop, Selected Papers*, volume 1074 of *Lecture Notes in Computer Science*, pages 185–200, 1995.

[81] Vincent van Oostrom. Confluence by decreasing diagrams, converted. In *Proc. 19th International Conference on Rewriting Techniques and Applications*, volume 5117 of *Lecture Notes in Computer Science*, pages 306–320, 2008.

[82] Ian Wehrman, Aaron Stump, and Eric M. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *Proc. 17th International Conference on Rewriting Techniques and Applications*, Lecture Notes in Computer Science, pages 287–296, 2006.

[83] Sarah Winkler and Aart Middeldorp. Termination tools in ordered completion. In *Proc. 5th International Joint Conference on Automated Reasoning*, volume 6173 of *Lecture Notes in Artificial Intelligence*, pages 518–532, 2010.

[84] Sarah Winkler, Haruhiko Sato, Aart Middeldorp, and Masahito Kurihara. Optimizing mkbtt (system description). In *Proc. 21st International Conference on Rewriting Techniques and Applications*, volume 6 of *Leipzig International Proceedings in Computer Science*, pages 373–384, 2010.

[85] Sarah Winkler, Haruhiko Sato, Aart Middeldorp, and Masahito Kurihara. Multi-completion with termination tools. *Journal of Automated Reasoning*, to appear.

[86] Harald Zankl, Bertram Felgenhauer, and Aart Middeldorp. CSI - A confluence tool. In *Proc. 23th International Conference on Automated Deduction*, volume 6803 of *Lecture Notes in Artificial Intelligence*, pages 499–505, 2011.

[87] Harald Zankl, Nao Hirokawa, and Aart Middeldorp. KBO orientability. *Journal of Automated Reasoning*, 43(2):173–201, 2009.

[88] Hans Zantema. Termination of term rewriting by semantic labelling. *Fundamenta Informaticae*, 24:89–105, 1995.

Index

Publications

in peer-reviewed international journals and conferences

(i) Dominik Klein and Nao Hirokawa: "Confluence of Non-Left-Linear TRSs via Relative Termination". *Proceedings of the 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning*, volume 7180 of *Lecture Notes in Computer Science*, pages 258–273, 2012.

(ii) Dominik Klein and Nao Hirokawa: "Maximal Completion". *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications*, volume 10 of *Leipzig International Proceedings in Informatics*, pages 71-80, 2011.

(iii) Dominik Klein, Frank G. Radmacher and Wolfgang Thomas: "Moving in a network under random failures: A complexity analysis". *Science of Computer Programming*, Article In Press.

(iv) Dominik Klein, Frank G. Radmacher and Wolfgang Thomas: "The Complexity of Reachability in Randomized Sabotage Games". *Fundamentals of Software Engineering, Revised Selected Papers*, volume 5961 of *Lecture Notes in Computer Science*, pages 162-177, 2009.

in domestic proceedings

(v) Dominik Klein: "Solving Randomised Sabotage Games for Navigation in Networks". *Informatiktage 2008. Fachwissenschaftlicher Informatik-Kongress*, volume S-6 of *Lecture Notes in Informatics*, pages 15-18, 2008.