Master's Thesis

Equational Reasoning by Maximal Ordered Completion

Saito Teppei

Supervisor Hirokawa Nao

Graduate School of Advanced Science and Technology
Japan Advanced Institute of Science and Technology
(Information Science)

March 2023

Abstract

This thesis studies equational theorem proving based on ordered completion. It is known that a reduction order is a critical parameter for ordered completion. To improve the capability of ordered completion, we present a new class of reduction orders, dubbed generalized weighted path orders. While the original weighted path order (Yamada et al. 2013) is a complete characterization of simplification orders including Knuth–Bendix orders and lexicographic path orders, the new class not only subsumes the original one, but also instantiates non-simplification orders. For instance, provided equational axioms for round-up division

$$\begin{aligned} \mathsf{0} - y &\approx \mathsf{0} & x - \mathsf{0} &\approx x \\ \mathsf{s}(x) - \mathsf{s}(y) &\approx x - y & & \\ \mathsf{0} \div \mathsf{s}(y) &\approx \mathsf{0} & \mathsf{s}(x) \div \mathsf{s}(y) &\approx \mathsf{s}((x - y) \div \mathsf{s}(y)) \end{aligned}$$

refutation of a non-theorem (e.g. $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$) by ordered completion fails if a simplification order is employed. In contrast, ordered completion with a generalized weighted path order is capable of giving refutation to such a non-theorem. However, the generalized weighted path order forces us to revisit a classical problem of ordered completion, namely selection of a reduction order: given an equational system, we do not know an appropriate reduction order in advance. The problem is even worse in the case of the generalized weighted path order since the powerfulness of the ordering relies on vast search space of its input parameters. As a solution to this problem, we integrate the generalized weighted path order into maximal ordered completion by Winkler and Moser (2018), in which we encode selection of a reduction order into MaxSMT constraints. Furthermore, to improve the efficiency, we propose a new variant of maximal ordered completion, which incorporates ordered completion without deduce, namely simplification, following the approach of Hirokawa (2021). The new ordering and the new ordered completion have been implemented in the equational theorem prover Toma, and experimental results show potential of our approach.

Contents

Chapter 1

Introduction

First we introduce equational reasoning, and its problems together with our approach. Next we give an overview of our contributions and the structure of the thesis.

1.1 Equational Reasoning

Throughout the thesis we discuss equational reasoning. In particular, we consider the word problem:

> instance: an equational system $\mathcal{E}$ and an equation $s \approx t$
>
> question: does $\mathcal{E}$ entail $s \approx t$?

For instance, consider the following equational system $\mathcal{E}_{-}$:

$$\begin{array}{llll} (1): & \mathsf{0} - y \approx \mathsf{0} & (2): & x - \mathsf{0} \approx x \\ (3): & \mathsf{s}(x) - \mathsf{s}(y) \approx x - y & & \end{array}$$

Here numbers are represented as Peano numbers: $\mathsf{0}$ denotes zero, $\mathsf{s}(\mathsf{0})$ denotes one, $\mathsf{s}(\mathsf{s}(\mathsf{0}))$ denotes two, and so on. So this system can be taken as an axiomatization of cut-off minus. Informally, we have $2 - 1 = 1$ but $1 - 2 = 0$ in this system. More formally, we can derive $\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}) \approx \mathsf{s}(\mathsf{0})$ and $\mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{0}$ by simply rewriting the left-hand sides to the right-hand sides using the axioms. For example, we can derive $\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}) \approx \mathsf{s}(\mathsf{0})$ as follows:

$$\begin{aligned} \mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}) &\approx \mathsf{s}(\mathsf{0}) - \mathsf{0} && \text{by (3)} \\ &\approx \mathsf{s}(\mathsf{0}) && \text{by (2)} \end{aligned}$$

Therefore, $\mathcal{E}_-$ entails $\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}) \approx \mathsf{s}(\mathsf{0})$. In other words, $\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}) \approx \mathsf{s}(\mathsf{0})$ is valid under the axioms. Similarly, we have $\mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{0}$ as follows:

$$\begin{aligned} \mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{s}(\mathsf{0})) &\approx \mathsf{0} - \mathsf{s}(\mathsf{0}) && \text{by (3)} \\ &\approx \mathsf{0} && \text{by (1)} \end{aligned}$$

So $\mathcal{E}_-$ entails $\mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{0}$.

Ordered completion [4] is a method to solve the word problem automatically. The approach is to transform an equational system into a *complete* set of rewrite rules, namely a complete term rewrite system (TRS). Once such a TRS $\mathcal{R}$ is obtained, the word problem on the original equational system is decidable; the complete TRS $\mathcal{R}$ decides the word problem in the following way:

1. Given an equation $s \approx t$, the procedure rewrites the terms s and t by using the rules in $\mathcal{R}$ from left to right, until no rule is applicable to the terms.

2. Let s' and t' be the terms obtained by the first step. If s' and t' are syntactically identical, then the problem is affirmatively solved. Otherwise, the problem is negatively solved.

For instance, by using ordered completion we obtain the following TRS $\mathcal{R}_-$ from the equational system $\mathcal{E}_-$:

$$\begin{aligned} \mathsf{0} - y &\to \mathsf{0} & \qquad x - \mathsf{0} &\to x \\ \mathsf{s}(x) - \mathsf{s}(y) &\to x - y \end{aligned}$$

There is no difference between $\mathcal{E}_-$ and $\mathcal{R}_-$, except for the separator symbols, $\approx$ and $\to$. Intuitively, the equations in $\mathcal{E}_-$ are not oriented, but the rules in $\mathcal{R}_-$ are oriented. This means that the direction of rewriting are restricted in $\mathcal{R}_-$ as opposed to $\mathcal{E}_-$, but the power of $\mathcal{R}_-$ is still sufficient to solve word problems on $\mathcal{E}_-$. This difference shows up when we consider an equation with the negative answer. Suppose that we are asked if $\mathcal{E}_-$ entails $\mathsf{0} \approx \mathsf{s}(\mathsf{0})$. We know that this equation is *wrong* by intuition, but how can we prove this? The difficulty comes from the fact that we can use equations in $\mathcal{E}_-$ either from left to right, or from right to left. For instance, we can rewrite $\mathsf{0}$ to $\mathsf{0} - \mathsf{s}(\mathsf{0})$ using the equation $\mathsf{0} - y \approx \mathsf{0}$ by instantiating y with $\mathsf{s}(\mathsf{0})$. How can we guarantee that $\mathsf{0}$ never reaches to $\mathsf{s}(\mathsf{0})$, even if we cleverly instantiate variables in equations? This is why the complete TRS $\mathcal{R}_-$ is helpful. We can prove that $\mathsf{0} \approx \mathsf{s}(\mathsf{0})$ does not hold in $\mathcal{E}_-$ using the decision procedure via the complete TRS $\mathcal{R}_-$.

1. We cannot rewrite $\mathsf{0}$ and $\mathsf{s}(\mathsf{0})$ using the rules in $\mathcal{R}_-$ from left to right.

2. The terms $\mathsf{0}$ and $\mathsf{s}(\mathsf{0})$ are not syntactically identical. Thus, the answer to the word problem $\mathsf{0} \approx \mathsf{s}(\mathsf{0})$ is negative.

This is how we can solve the word problem using ordered completion; it tries to find a complete TRS that gives a decision procedure. In the subsequent sections we introduce two problems in ordered completion, namely

- limitation of simplification orders and
- selection of a reduction order,

together with our approach to these problems.

1.2 Limitation of Simplification Orders

Let us give a further detail of ordered completion. In addition to an equational system $\mathcal{E}$ and a goal, the procedure takes an ordering $>$ on terms, which is called a *reduction order*. Informally, the reduction order $>$ determines the proof strategy, such as how equations are oriented. Consider the following equational system $\mathcal{E}_\div$, made of $\mathcal{E}_-$ plus two additional equations:

$$
\begin{array}{llll}
(1): & \mathsf{0} - y \approx \mathsf{0} & (2): & x - \mathsf{0} \approx x \\
(3): & \mathsf{s}(x) - \mathsf{s}(y) \approx x - y & & \\
(4): & \mathsf{0} \div \mathsf{s}(y) \approx \mathsf{0} & (5): & \mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{array}
$$

This system axiomatizes round-up division. Informally we have $2 \div 2 = 1$ but $3 \div 2 = 2$. Formally we have the following derivation of $\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$:

$$
\begin{array}{rll}
\underline{\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))} & \approx \mathsf{s}(\underline{(\mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) & \text{by (5)} \\
& \approx \mathsf{s}(\underline{(\mathsf{0} - \mathsf{0})} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) & \text{by (3)} \\
& \approx \mathsf{s}(\underline{\mathsf{0} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))}) & \text{by (1)} \\
& \approx \mathsf{s}(\mathsf{0}) & \text{by (4)}
\end{array}
$$

Similarly we have the following derivation of $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{s}(\mathsf{0}))$:

$$
\begin{array}{rll}
\underline{\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))} & \approx \mathsf{s}(\underline{(\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) & \text{by (5)} \\
& \approx \mathsf{s}(\underline{(\mathsf{s}(\mathsf{0}) - \mathsf{0})} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) & \text{by (3)} \\
& \approx \mathsf{s}(\underline{\mathsf{s}(\mathsf{0}) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))}) & \text{by (2)} \\
& \approx \mathsf{s}(\mathsf{s}(\underline{(\mathsf{0} - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0})))) & \text{by (5)} \\
& \approx \mathsf{s}(\mathsf{s}(\underline{\mathsf{0} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))})) & \text{by (1)} \\
& \approx \mathsf{s}(\mathsf{s}(\mathsf{0})) & \text{by (4)}
\end{array}
$$

Observe that the derivations above use equations only from left to right. As we expect, we can obtain the following complete TRS by orienting the equations from left to right:

$$\begin{aligned}
\mathsf{0} - y &\to \mathsf{0} & x - \mathsf{0} &\to x \\
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
\mathsf{0} \div \mathsf{s}(y) &\to \mathsf{0} & \mathsf{s}(x) \div \mathsf{s}(y) &\to \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}$$

The TRS gives a decision procedure for the word problem of $\mathcal{E}_{\div}$. For example, the answer to $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{s}(\mathsf{0}))$ is affirmative: by rewriting the left-hand side and the right-hand side to maximum, we obtain the same term $\mathsf{s}(\mathsf{s}(\mathsf{0}))$. For another example, the answer to $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ is negative: by rewriting the left-hand side and the right-hand side to maximum, we obtain the different terms $\mathsf{s}(\mathsf{s}(\mathsf{0}))$ and $\mathsf{s}(\mathsf{0})$.

Unfortunately, current equational theorem provers based on ordered completion cannot perform the inferences shown above. This is not a problem of their implementations, but a problem of reduction orders they use. To obtain the complete TRS, we need to supply ordered completion with a reduction order satisfying the following orientation:

$$\begin{aligned}
\mathsf{0} - y &> \mathsf{0} & x - \mathsf{0} &> x \\
\mathsf{s}(x) - \mathsf{s}(y) &> x - y \\
\mathsf{0} \div \mathsf{s}(y) &> \mathsf{0} & \mathsf{s}(x) \div \mathsf{s}(y) &> \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{aligned}$$

But it is known that this orientation is impossible for the class of reduction orders, so-called the *simplification order*. The class subsumes the Knuth–Bendix order [19] and the lexicographic path order [17], which most implementations only support. At best a run of ordered completion with a simplification order orients $\mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))$ from right to left. As a result the run *diverges*, generating infinitely many rewrite rules:

$$\begin{aligned}
\mathsf{0} - y &\to \mathsf{0} & x - \mathsf{0} &\to x \\
\mathsf{s}(x) - \mathsf{s}(y) &\to x - y \\
\mathsf{0} \div \mathsf{s}(y) &\to \mathsf{0} & \mathsf{s}(x) \div \mathsf{s}(y) &\leftarrow \mathsf{s}((x - y) \div \mathsf{s}(y)) \\
\mathsf{s}(\mathsf{0}) \div \mathsf{s}(y) &\to \mathsf{s}(\mathsf{0}) & \mathsf{s}(\mathsf{s}(x)) \div \mathsf{s}(\mathsf{s}(y)) &\leftarrow \mathsf{s}((x - y) \div \mathsf{s}(\mathsf{s}(y))) \\
\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(y)) &\to \mathsf{s}(\mathsf{0}) & &\vdots \\
\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{s}(y))) &\to \mathsf{s}(\mathsf{0}) & &\vdots \\
&\vdots & &\vdots
\end{aligned}$$

Still the run answers affirmatively to $\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ and $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{s}(\mathsf{0}))$ using generated rewrite rules such as $\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}((y))) \to \mathsf{s}(\mathsf{0})$ and $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{s}((y)))) \to \mathsf{s}(\mathsf{0})$ respectively, paying the cost of producing auxiliary rules. Even worse is that it is unable to answer negatively to $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ as an infinite TRS cannot be used to give a refutation automatically. As we can see in the example above, a class of reduction orders used in ordered completion can restrict its power. Therefore, it is reasonable to develop reduction orders located beyond the realm of the simplification order.

In this thesis, we develop an extension of the weighted path order (WPO) [34], dubbed the generalized weighted path order (GWPO). As well as the WPO, the GWPO takes two parameters: a precedence as a syntactic component, and an algebra as a semantic component. In contrast to the original WPO, algebras of the GWPO need not be *simple*. Thanks to the flexibility of algebras, the GWPO is located beyond the realm of the simplification order, and now we can orient $\mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))$ from left to right. However, this give rise to another problem, selection of a reduction order.

1.3 Selection of a Reduction Order

Selection of a reduction order is a critical factor that determines success of ordered completion. Let us revisit the example of round-up division. We can choose parameters so that the GWPO $>_{\mathsf{gwpo}}$ orients $\mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))$ from left to right.

$$\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{gwpo}} \mathsf{s}((x - y) \div \mathsf{s}(y))$$

As we see in the previous section, a run with the reduction order $>_{\mathsf{gwpo}}$ eventually generates the complete TRS that gives a decision procedure for the word problem. However, the GWPO is so flexible to be able to orient the equation from right to left, depending on a choice of a precedence and an algebra.

$$\mathsf{s}((x - y) \div \mathsf{s}(y)) >_{\mathsf{gwpo}} \mathsf{s}(x) \div \mathsf{s}(y)$$

If we choose such a reduction order $>_{\mathsf{gwpo}}$, then the run diverges, and we are not able to give the negative answer to a problem such as $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$. As we can see in the example, choice of a reduction order matters in ordered completion. But how can we know which reduction order gives a complete TRS in advance? In fact, selection of a reduction order has been an open problem in theorem proving research [27, Open problem 3].

An approach to this problem is maximal completion [18]. In maximal completion, selection of a reduction order is encoded as a constraint solving problem, which reflects high-level heuristics. Later a variant of maximal completion was developed, maximal ordered completion by Winkler and Moser [33]. They also developed the equational theorem prover MædMax, based on maximal ordered completion. As high-level heuristics, MædMax focuses on reducibility; it tries to find a reduction order that maximizes the power to rewrite the goal and the equations, namely the reducibility of the rewrite system.

To unleash the potential of the GWPO in ordered completion, we employ maximal ordered completion. Unlike typical simplification orders, such as the Knuth–Bendix order and the lexicographic path order, the parameters of the GWPO is highly flexible, which results in a vast search space. Therefore, it is reasonable to find actual parameters of an ordering by encoding high-level heuristics (e.g. maximization of reducibility) as a constraint solving problem, rather than using a fixed strategy. For instance, the state-of-the-art ordered completion tool Twee uses the Knuth–Bendix order with a fixed strategy [30], which does not pay off when we use the GWPO. Moreover, we integrate a new variant of maximal completion [13] into maximal ordered completion. This allows us to eliminate redundant equations using a variant of the Knuth–Bendix completion [19], namely *simplification*, which is not allowed in the original maximal ordered completion. The integration of simplification enhances performance on problems which contain numerous equations.

1.4 An Outline and Contributions

The thesis is organized as follows: Chapter 2 serves as preliminaries. Chapter 3 introduces the generalized weighted path order. Chapter 4 introduces a new theorem proving procedure, maximal ordered completion with simplification. Chapter 5 concludes the thesis. The contributions of this thesis are given in the following paragraphs:

The generalized weighted path order. We introduce a new class of reduction orders, the generalized weighted path order (GWPO), which gives a way to construct non-simplification orders. The relation between the GWPO and existing classes of reduction orders, including the weighted path order (WPO), the Knuth–Bendix order (KBO) and the lexicographic path order (LPO), are shown together with experimental results.

Maximal ordered completion with simplification. We introduce a new equational theorem proving procedure, which integrates simplification into maximal ordered completion [33] by following the existing approach [13] for the standard completion. A correctness proof of the procedure is given. Also, the performance is evaluated in various settings, including cases of the GWPO.

Toma. The reduction orders and theorem proving procedures proposed in the thesis have been implemented in the equational theorem prover Toma, available at:

`https://www.jaist.ac.jp/project/maxcomp/`

Settings for experiments are accessible via command line options. The thesis uses Toma 0.3.

Chapter 2

Preliminaries

Throughout the thesis, we assume familiarity with term rewriting [3].

2.1 Abstract Rewriting

Definition 1 (proper orders)**.** Let $>$ be a binary relation on a set X. We call $>$ a *proper order* if $>$ satisfies irreflexivity and transitivity:

- irreflexivity: $x \not> x$ for all $x \in X$
- transitivity: for all $x, y, z \in X$, if $x > y$ and $y > z$, then $x > z$

Definition 2 (well-foundedness)**.** We call a binary relation R is *well-founded* if there exists no infinite sequence $x_0 \mathrel{R} x_1 \mathrel{R} x_2 \ldots$. A set equipped with a well-founded proper order is called a *well-founded set.*

We recall two ways to extend well-founded orders, lexicographic extension and multiset extension.

Definition 3 (lexicographic extension)**.** Let A be a set equipped with a proper order $>$ and let $n \in \mathbb{N}$. For elements $a_1, \ldots, a_n, b_1, \ldots, b_n \in A$ we write $(a_1, \ldots, a_n) >^{\mathsf{lex}} (b_1, \ldots, b_n)$ if $a_1 = b_1, \ldots, a_{k-1} = b_{k-1}$ and $a_k > b_k$ for some $k \in \{1, \ldots, n\}$. The relation $>^{\mathsf{lex}}$ on A^n is called the *lexicographic extension* of $>$.

Proposition 4. *Let $n \in \mathbb{N}$. If $(A, >)$ is a well-founded set, so is $(A^n, >^{\mathsf{lex}})$.*

Definition 5 (multisets)**.** Let X be a set. A function $M : X \to \mathbb{N}$ is called a *multiset* on X if there is only finitely many elements $x \in X$ with $M(x) > 0$. The set of all multisets on X is denoted by $\mathcal{M}(X)$. Next we extend the notions for sets $\in, \subseteq, \varnothing, \cup$ and $\setminus$ to multisets. Let $M_1, M_2 \in \mathcal{M}(X)$. We write $x \in M_1$ if $M_1(x) > 0$. We also write $M_1 \subseteq M_2$ if $M_1(x) \leq M_2(x)$ for all $x \in X$. The *empty multiset* $\varnothing$ is the constant function: $\varnothing(x) = 0$ for all $x \in X$. The *union* $M_1 \cup M_2$ is the multiset defined by the point-wise sum: $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$ for all $x \in X$. The *difference* $M_1 \setminus M_2$ is the multiset defined by cut-off minus:

$$(M_1 \setminus M_2)(x) = \begin{cases} M_1(x) - M_2(x) & \text{if } M_1(x) \geq M_2(x) \\ 0 & \text{otherwise} \end{cases}$$

for all $x \in X$.

Definition 6. Let X be a set equipped with a proper order $>$, and let $M_1, M_2 \in \mathcal{M}(X)$. We write $M_1 >^{\mathsf{mul}} M_2$ if there exists multisets M_3, M_4 that satisfies the following conditions:

- $M_3 \neq \varnothing$ and $M_3 \subseteq M_1$.
- For every $x \in M_4$, there exists $y \in M_3$ with $y > x$.
- $M_2 = (M_1 \setminus M_3) \cup M_4$.

Proposition 7. *If $(X, >)$ is a well-founded set, so is $(\mathcal{M}(X), >^{\mathsf{mul}})$.*

Definition 8 (abstract rewrite systems)**.** A set A equipped with a binary relation $\to$ is called an *abstract rewrite system (ARS)*. The symbol $\mathcal{A}$ is used for the ARS $(A, \to)$. We write $\to_1 \cdot \to_2$ for the composition of relations $\to_1$ and $\to_2$. We introduce the following notations:

- $\leftarrow$ denotes the inverse relation of $\to$.
- $\to^=$ denotes the symmetric closure of $\to$.
- $\to^*$ denotes the symmetric and transitive closure of $\to$.
- $\to^+$ denotes the transitive closure of $\to$.
- $\leftrightarrow$ denotes the reflexive closure of $\to$.
- $\leftrightarrow^*$ denotes the symmetric, transitive, and reflexive closure of $\to$. A sequence $s_0 \leftrightarrow s_1 \leftrightarrow \ldots \leftrightarrow s_n$ is called a *conversion* between s_0 and s_n.
- We write $a \downarrow b$ if $a \to^* \cdot {}^*\!\leftarrow b$, and we say that a and b are *joinable.*
- An element $a \in A$ that has no element $b \in A$ with $a \to b$ is called a *normal form.* The set of all normal forms is denoted by $\mathsf{NF}(\mathcal{A})$. If $a \to^* b \in \mathsf{NF}(\mathcal{A})$, we say b is a normal form of a, or equivalently a has a normal form b.

We are ready to define basic properties of abstract rewrite systems.

- The ARS is *terminating* if $\to$ is well-founded.
- The ARS is *confluent* if the inclusion ${}^*\!\leftarrow \cdot \to^* \subseteq \downarrow$ holds.
- The ARS is *complete* if it is terminating and confluent.
- The ARS is *locally confluent* if the inclusion $\leftarrow \cdot \to \subseteq \downarrow$ holds.
- The ARS has *the Church–Rosser property* if the inclusion $\leftrightarrow^* \subseteq \downarrow$ holds.

Confluence and the Church–Rosser property coincide:

Proposition 9. *Every ARS is confluent if and only if it has the Church–Rosser property.*

Proposition 10 (Newman's Lemma [26])**.** *Every terminating and locally confluent ARS is confluent.*

If an ARS is confluent, every element has unique normal forms. So the normal form of an element a is unambiguously denoted by $a\!\downarrow$ provided that a has a normal form.

Proposition 11. *Let $\mathcal{A}$ be a confluent ARS, and a be an element of $\mathcal{A}$. If a has normal forms b, c, then $b = c$.*

2.2 Term Rewriting

Definition 12 (terms)**.** The set of *variables* $\mathcal{V}$ is an arbitrary infinite set. The *signature* $\mathcal{F}$ is an arbitrary set such that a unique natural number n is given to each $f \in \mathcal{F}$. The number n is called the *arity* of f. An element in $\mathcal{F}$ is called a *function symbol.* A function symbol with the arity 0 is called a *constant.* The set $\mathcal{T}(\mathcal{F}, \mathcal{V})$ of all *terms* is defined inductively:

- Every variable is a term.
- Let n be a natural number, f a function symbol with the arity n, and $t_1, \ldots, t_n$ terms. Then $f(t_1, \ldots, t_n)$ is a term.

Note that the set of variables $\mathcal{V}$ and the signature $\mathcal{F}$ are fixed when terms are involved. The set of variables appears in a term t is denoted by $\mathsf{Var}(t)$. The number of occurrence of a variable x in a term t is denoted by $|t|_x$. A term that has no variable is called a *ground* term. For a non-variable term $s = f(s_1, \ldots, s_n)$, the function symbol f is called the *root symbol* of s, denoted by $\mathsf{root}(s)$.

Definition 13 (substitutions)**.** A *substitution* σ is a function from $\mathcal{V}$ to $\mathcal{T}(\mathcal{F},\mathcal{V})$ that has only finitely many variables $x \in \mathcal{V}$ satisfying $\sigma(x) \neq x$. The *application* $t\sigma$ of a substitution σ to a term t is defined by recursion:

- $x\sigma = \sigma(x)$ for each variable x.
- $\sigma(f(t_1,\dots,t_n)) = f(t_1\sigma,\dots,t_n\sigma)$ for all n-ary function symbols f and terms $t_1,\dots,t_n$.

A *variable substitution* σ is a substitution such that $\sigma(x) \in \mathcal{V}$ for all $x \in \mathcal{V}$. A bijective variable substitution is called a *renaming.*

Definition 14 (ground substitutions)**.** Let s and t be terms. A substitution σ is a *ground substitution* for s and t if $s\sigma$ and $t\sigma$ are ground. If the terms s and t are clear from the context, we may simply say that σ is a ground substitution.

Definition 15 (contexts)**.** The *hole* $\square$ is a fresh constant symbol with $\square \notin \mathcal{F}$. A context C is a term that contains exactly one hole. For each term t and context C the term $C[t]$ is defined by recursion:

- $x[t] = x$ for all variables x
- $\square[t] = t$
- $f(t_1,\dots,t_n)[t] = f(t_1[t],\dots,t_n[t])$ for all n-ary function symbols f and terms $t_1,\dots,t_n$

Definition 16 (positions, subterms)**.** Let t be a term. The set of all *positions* $\mathsf{Pos}(t)$ of t is a set of strings over $\mathbb{N}$, defined by recursion:

- $\mathsf{Pos}(x) = \{\epsilon\}$ for all variables x.
- $\mathsf{Pos}(f(t_1,\dots,t_n)) = \{\epsilon\} \cup \bigcup_{k=0}^{n} \{k \cdot p \mid p \in \mathsf{Pos}(t_k)\}$ for each function symbol f with arity n and terms $t_1,\dots,t_n$.

The *subterm* $t|_p$ of a term t at a position $p \in \mathsf{Pos}(t)$ is defined by recursion $t|_\epsilon = t$ and $f(t_1,\dots,t_n)|_{k \cdot p} = t_k|_p$. We write $s \trianglerighteq t$ if t is a subterm of s at some position p. In particular, we write $s \triangleright t$ if $p \neq \epsilon$, and call t a *proper subterm* of s. We write $t[]_p$ for the context obtained from t by replacing the subterm $t|_p$ by the hole. Formally, we define $t[]_p$ by recursion $t[]_\epsilon = \square$ and $f(t_1,\dots,t_n)[]_{k \cdot p} = f(t_1,\dots,t_k[]_p,\dots,t_n)$. We write $t[u]_p$ for $(t[]_p)[u]$.

Definition 17 (rewrite relations/rewrite orders/reduction orders)**.** Let R be a binary relation on terms.

- R is *closed under substitutions* if $s\sigma \; R \; t\sigma$ for all substitutions σ and terms s, t with $s \; R \; t$.
- R is *closed under contexts* if $C[s] \; R \; C[t]$ for all contexts C and terms s, t with $s \; R \; t$.
- R is a *rewrite relation* if it is closed under substitutions and contexts.
- R is a *rewrite order* if it is a proper order and a rewrite relation.
- R is a *reduction order* if it is a well-founded rewrite order.

Definition 18 (encompassment)**.** A term s *encompasses* a term t if $s = C[t\sigma]$ for some context C and substitution σ, and we write $s \trianglerighteq t$. The *strict encompassment* $\vartriangleright$ is the difference $\trianglerighteq \setminus \trianglerighteq^{-1}$.

Proposition 19 ([14, Lemma 2.3])**.** *If R is a well-founded rewrite relation then $\trianglerighteq^* \cdot (R \cup \vartriangleright) \cdot \trianglerighteq^*$ is well-founded.*

Definition 20 (algebra)**.** Let $\mathcal{F}$ be a signature. An $\mathcal{F}$-*algebra* $\mathcal{A}$ consists of

- a set A (carrier) and
- an interpretation $f_{\mathcal{A}} : A^n \to A$ for each $f^{(n)} \in \mathcal{F}$.

A function $\alpha : \mathcal{V} \to A$ is called an *assignment* for the algebra $\mathcal{A}$. The assignment α is extended to the *valuation* $[\alpha] : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to A$ by recursion:

- $[\alpha]_{\mathcal{A}}(x) = \alpha(x)$ for each $x \in \mathcal{V}$.
- $[\alpha]_{\mathcal{A}}(f(t_1, \ldots, t_n)) = f_{\mathcal{A}}([\alpha]_{\mathcal{A}}(t_1), \ldots, [\alpha]_{\mathcal{A}}(t_n))$ for all $f^{(n)} \in \mathcal{F}$ and terms $t_1, \ldots, t_n$.

Definition 21 (well-foundedness, simplicity and weak monotonicity of algebras)**.** Let $\mathcal{A}$ be an algebra equipped with a proper order $>$ on its carrier A. Here $\geq$ denotes the reflexive closure of $>$.

- The algebra $\mathcal{A}$ is *well-founded* if $>$ is well-founded.
- The algebra $\mathcal{A}$ is *simple* if

  $$f_{\mathcal{A}}(a_1, \ldots, a_n) \geq a_i$$

  for all $f^{(n)} \in \mathcal{F}$, and $a_1, \ldots, a_n \in A$, and $i \in \{1, \ldots, n\}$.

- The algebra $\mathcal{A}$ is *weakly monotone* if
$$f_{\mathcal{A}}(a_1, \ldots, a_i, \ldots, a_n) \geq f_{\mathcal{A}}(a_1, \ldots, b, \ldots, a_n)$$
for all $f^{(n)} \in \mathcal{F}$, and $a_1, \ldots, a_n, b \in A$, and $i \in \{1, \ldots, n\}$ with $a_i > b$.
- The algebra $\mathcal{A}$ is *simple monotone* if $\mathcal{A}$ is simple and weakly monotone.

Definition 22. Let $\mathcal{A}$ be an algebra equipped with proper order $>$. We define the binary relations $>_{\mathcal{A}}$, $\geq_{\mathcal{A}}$ and $=_{\mathcal{A}}$ as follows:

- $s >_{\mathcal{A}} t$ if $[\alpha](s) > [\alpha](t)$ for all assignments α.
- $s \geq_{\mathcal{A}} t$ if $[\alpha](s) \geq [\alpha](t)$ for all assignments α.
- $s =_{\mathcal{A}} t$ if $[\alpha]_{\mathcal{A}}(s) = [\alpha]_{\mathcal{A}}(t)$ for all assignments α.

Definition 23 (equational systems)**.** A pair (s, t) of terms is called an *equation*, denoted by $s \approx t$. An *equational system (ES)* $\mathcal{E}$ is a set of equations. We write $\mathcal{E}^{-1}$ for the set $\{t \approx s \mid s \approx t \in \mathcal{E}\}$. We write $s \rightarrow_{\mathcal{E}} t$ if there exist an equation $\ell \approx r \in \mathcal{E}$, a context C, and a substitution σ such that $s = C[\ell\sigma]$ and $t = C[r\sigma]$. An algebra $\mathcal{A}$ is a *model* of $\mathcal{E}$ if $\ell =_{\mathcal{A}} r$ for all $\ell \approx r \in \mathcal{E}$. An equation $s \approx t$ is *valid* in $\mathcal{E}$ if $s =_{\mathcal{A}} t$ for all models $\mathcal{A}$ of $\mathcal{E}$.

Proposition 24 (Birkhoff's completeness theorem)**.** *Let $\mathcal{E}$ be an equational system. An equation $s \approx t$ is valid in $\mathcal{E}$ if and only if $s \leftrightarrow^*_{\mathcal{E}} t$.*

Definition 25 (the word problem)**.** The *word problem* is given in the following scheme:

instance: an equational system $\mathcal{E}$ and an equation $s \approx t$

question: does $\mathcal{E}$ entail $s \approx t$?

Example 26. Now we are ready to revisit an example in the introduction with formal knowledge of rewriting. We consider word problems under the following equational system $\mathcal{E}_{\div}$ for round-up division:

$$\begin{array}{llll}
(1): & \mathsf{0} - y \approx \mathsf{0} & (2): & x - \mathsf{0} \approx x \\
(3): & \mathsf{s}(x) - \mathsf{s}(y) \approx x - y & & \\
(4): & \mathsf{0} \div \mathsf{s}(y) \approx \mathsf{0} & (5): & \mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))
\end{array}$$

The equation $\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ is valid in $\mathcal{E}_{\div}$ as we admit the following conversion:

$$\begin{aligned}
\underline{\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))} &\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{(\mathsf{s}(\mathsf{0}) - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) && \text{by (5)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{(\mathsf{0} - \mathsf{0})} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) && \text{by (3)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{\mathsf{0} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))}) && \text{by (1)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\mathsf{0}) && \text{by (4)}
\end{aligned}$$

Similarly we have the following conversion, which proves the validity of $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{s}(\mathsf{0}))$:

$$\begin{aligned}
\underline{\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))} &\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{(\mathsf{s}(\mathsf{s}(\mathsf{0})) - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) && \text{by (5)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{(\mathsf{s}(\mathsf{0}) - \mathsf{0})} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))) && \text{by (3)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\underline{\mathsf{s}(\mathsf{0}) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))}) && \text{by (2)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\mathsf{s}(\underline{(\mathsf{0} - \mathsf{s}(\mathsf{0}))} \div \mathsf{s}(\mathsf{s}(\mathsf{0})))) && \text{by (5)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\mathsf{s}(\underline{\mathsf{0} \div \mathsf{s}(\mathsf{s}(\mathsf{0}))})) && \text{by (1)}\\
&\to_{\mathcal{E}_{\div}} \mathsf{s}(\mathsf{s}(\mathsf{0})) && \text{by (4)}
\end{aligned}$$

On the other hand, the system has a semantic counterpart of round-up division, namely a model. Define an algebra $\mathcal{A}$ as follows: set $\mathbb{N}$ as its carrier, and let $\mathsf{0}_{\mathcal{A}} = 0$, $\mathsf{s}_{\mathcal{A}}(x) = x + 1$, $-_{\mathcal{A}}$ cut-off minus, and $\div_{\mathcal{A}}$ round-up division. More formally, $-_{\mathcal{A}}$ and $\div_{\mathcal{A}}$ are defined as follows:

$$x -_{\mathcal{A}} y = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{otherwise} \end{cases}$$

$$x \div_{\mathcal{A}} y = \begin{cases} \lceil x/y \rceil & \text{if } y > 0 \\ 0 & \text{otherwise} \end{cases}$$

Here $\lceil x \rceil$ denotes the smallest natural number greater than a real number x. It is easy for us to see that $\mathcal{A}$ is a model of $\mathcal{E}_{\div}$, and we have $\mathsf{s}(\mathsf{s}(\mathsf{0})) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) =_{\mathcal{A}} \mathsf{s}(\mathsf{0})$ and $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) =_{\mathcal{A}} \mathsf{s}(\mathsf{s}(\mathsf{0}))$ from the completeness theorem. In particular, models are useful for disproofs. For instance, the equation $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ is invalid, as $\mathcal{A}$ interprets the left-hand side as 2 but the right-hand side as 1. This also means that there is no conversion between $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))$ and $\mathsf{s}(\mathsf{0})$, thanks to the completeness theorem. However, disproof via a model is difficult to automate because generally it is not easy to find a model, and even verify that it is a model: suppose that we are asked to prove that $\mathcal{A}$ is a model of $\mathcal{E}_{\div}$. We need to verify $\mathsf{s}(x) \div \mathsf{s}(y) =_{\mathcal{A}} \mathsf{s}((x - y) \div \mathsf{s}(y))$, but this requires $\lceil (x + 1)/(y + 1) \rceil = 1 + \lceil (x -_{\mathcal{A}} y)/(y + 1) \rceil$, which is a lemma to be found and proved automatically.

Notion of term rewrite systems plays an important role in mechanically solving the word problem. In particular, *termination* and *confluence* of term rewrite systems are of special interest, as they guarantee decidability of the word problem.

Definition 27 (term rewrite systems)**.** An equation $\ell \approx r$ is a *rule* if $\ell \notin \mathcal{V}$ and $\mathsf{Var}(r) \subseteq \mathsf{Var}(\ell)$. An equational system $\mathcal{R}$ is a *term rewrite system (TRS)* if every equation in $\mathcal{R}$ is a rule. A term rewrite system is terminating/confluent/complete if $\to_{\mathcal{R}}$ is terminating/confluent/complete.

Example 28 (continued from Example 26)**.** The equational system $\mathcal{E}_{\div}$ is a term rewrite system, so we also write $\mathcal{R}_{\div}$ for $\mathcal{E}_{\div}$ in subsequent examples. Moreover, $\mathcal{R}_{\div}$ is actually terminatinjg even though we postpone the proof of termination until Chapter 3.

How can we prove termination of a term rewrite system? A major method is finding a suitable reduction order for a term rewrite system.

Definition 29. A term rewrite system $\mathcal{R}$ is *compatible* with a reduction order $>$ if $\mathcal{R} \subseteq >$.

Proposition 30. *A term rewrite system is terminating if and only if it is compatible with some reduction order.*

The lexicographic path order is a well-known class of reduction orders.

Definition 31 (precedence)**.** A proper order on a signature is called a *precedence.*

Definition 32 (lexicographic path order)**.** Let $\succ$ be a precedence. We define the binary relation $>_{\mathsf{lpo}}$ on terms as the smallest relation satisfying the following conditions:

(subterm) $f(s_1, \ldots, s_n) >_{\mathsf{lpo}} t$ if $s_i >_{\mathsf{lpo}}^{=} t$ for some term s_i.

(lex) $f(s_1, \ldots, s_n) >_{\mathsf{lpo}} f(t_1, \ldots, t_n)$ if

- $(s_1, \ldots, s_n) >_{\mathsf{lpo}}^{\mathsf{lex}} (t_1, \ldots, t_n)$, and
- $f(s_1, \ldots, s_n) >_{\mathsf{lpo}} t_j$ for all $j \in \{1, \ldots, n\}$.

(precedence) $f(s_1, \ldots, s_n) >_{\mathsf{lpo}} g(t_1, \ldots, t_m)$ if

- $f \succ g$, and
- $f(s_1, \ldots, s_n) >_{\mathsf{lpo}} t_j$ for all $j \in \{1, \ldots, n\}$.

Informally $>_{\mathsf{lpo}}$ is given by the following inference system:

$$\frac{\exists i(s_i >_{\mathsf{lpo}}^{=} t)}{f(s_1,\ldots,s_n) >_{\mathsf{lpo}} t}\ \text{(subterm)}$$

$$\frac{(s_1,\ldots,s_n) >_{\mathsf{lpo}}^{\mathsf{lex}} (t_1,\ldots,t_n) \qquad \forall j(s >_{\mathsf{lpo}} t_j)}{s = f(s_1,\ldots,s_n) >_{\mathsf{lpo}} f(t_1,\ldots,t_n) = t}\ \text{(lex)}$$

$$\frac{f \succ g \qquad \forall j(s >_{\mathsf{lpo}} t_j)}{s = f(s_1,\ldots,s_n) >_{\mathsf{lpo}} g(t_1,\ldots,t_m) = t}\ \text{(precedence)}$$

The inference system is especially convenient when we give a proof of $s >_{\mathsf{lpo}} t$ for given terms s, t.

Definition 33. A rewrite order $>$ is a *simplification order* if it has the subterm property: $s > t$ for all terms $s \rhd t$. A TRS is *simply terminating* if it is compatible with some simplification order.

The following proposition is well-known, which is useful to establish well-foundedness of reduction orders. For a reference, see e.g. [3].

Proposition 34. *Every simplification order over a finite signature is well-founded.*

In this thesis we only consider finite signatures. Thus simplification orders in this thesis are reduction orders.

Proposition 35 ([17])**.** *Every instance of the lexicographic path order is a simplification order.*

We give another class of the simplification order, the Knuth–Bendix order [19].

Definition 36 (weights)**.** A *weight* (w, w_0) is a pair of a function $w : \mathcal{F} \to \mathbb{N}$ and a natural number w_0 such that $w(c) \geq w_0$ for all constants c. Given a weight (w, w_0), the weight $w(t)$ of a term t is defined by recursion:

- $w(x) = w_0$ for all variables x.
- $w(f(t_1,\ldots,t_n)) = w(f) + \sum_{k=0}^{n} w(t_k)$ for all n-ary function symbols f and terms $t_1,\ldots,t_n$.

Definition 37 ([19])**.** Given a weight (w, w_0) and a precedence $\succ$, the *Knuth–Bendix order* $>_{\mathsf{kbo}}$ is the smallest relation such that $s >_{\mathsf{kbo}} t$ if and only if $|s|_x \geq |t|_x$ for all variables x, $w(s) \geq w(t)$, and one of the following conditions holds:

- $w(s) > w(t)$
- t is a variable and $s = f^n(t)$ for some unary $f \in \mathcal{F}$ and positive natural number n
- $f \succ g$ where s and t are in the form of $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_m)$
- $(s_1, \ldots, s_n) >^{\mathsf{lex}}_{\mathsf{kbo}} (t_1, \ldots, t_n)$ where s and t are in the form of $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_n)$

The Knuth–Bendix order is *admissible* if f is maximum with respect to the precedence $\succ$ for all unary function symbols f with $w(f) = 0$.

Proposition 38 ([19])**.** *Every admissible Knuth–Bendix order is a simplification order.*

Definition 39 (the embedding rules)**.** Let $\mathcal{F}$ be a signature. The TRS $\mathcal{E}\mathsf{mb}$ consists of rules $f(x_1, \ldots, x_n) \to x_i$ for all $f^{(n)} \in \mathcal{F}$ and $i \in \{1, \ldots, n\}$.

Proposition 40 ([25, Lemma 4.6])**.** *Let $\mathcal{R}$ be a TRS and $\mathcal{E}\mathsf{mb}$ a simplification order. The following statements are equivalent:*

- *The TRS $\mathcal{R}$ is simply terminating.*
- *The TRS $\mathcal{R} \cup \mathcal{E}\mathsf{mb}$ has no cycle: i.e. $t \to^+_{\mathcal{R}\cup\mathcal{E}\mathsf{mb}} t$ for no term t.*

This characterization is useful when we show that a TRS $\mathcal{R}$ is not simply terminating; it suffices to give a cyclic rewrite sequence in $\mathcal{R} \cup \mathcal{E}\mathsf{mb}$.

Example 41 (continued from Example 28)**.** The TRS $\mathcal{R}_\div$ is not simply terminating because $\mathcal{R}_\div \cup \mathcal{E}\mathsf{mb}$ has a cycle:

$$\begin{aligned} \mathsf{s}(x) \div \mathsf{s}(\mathsf{s}(x)) &\to_{\mathcal{R}_\div} \mathsf{s}((x - \mathsf{s}(x)) \div \mathsf{s}(\mathsf{s}(x))) \\ &\to_{\mathcal{E}\mathsf{mb}} (x - \mathsf{s}(x)) \div \mathsf{s}(\mathsf{s}(x)) \\ &\to_{\mathcal{E}\mathsf{mb}} \mathsf{s}(x) \div \mathsf{s}(\mathsf{s}(x)) \end{aligned}$$

Therefore, the TRS is not compatible with the KBO nor the LPO.

For confluence of a term rewrite system, we have a useful notion, namely critical pairs. It is well-known that confluence of a terminating term rewrite system is characterized by its critical pairs.

Definition 42 (most general unifiers)**.** A substitution ρ is a *unifier* for s and t if $s\rho = t\rho$. In particular, ρ is a *most general unifier* (called a *mgu* for short) if for all unifiers σ for s and t, there exists τ such that $\rho\tau = \sigma$. Here, $\rho\tau$ is the substitution given by $(\rho\tau)(x) = (x\rho)\tau$ for all variables x.

Definition 43 (variants)**.** Equations $\ell_1 \approx r_1$ and $\ell_2 \approx r_2$ are *variants* if there is a renaming σ such that $\ell_1\tau = \ell_2$ and $r_1\tau = r_2$.

Definition 44 (critical overlap/peak/pair)**.** Let $\mathcal{R}$ be a TRS, and $\ell_1 \to r_1, \ell_2 \to r_2$ rules, and $p \in \mathsf{Pos}(\ell_2)$. The triple $(\ell_1 \to r_1, p, \ell_2 \to r_2)$ is a *critical overlap* if all the following conditions are satisfied:

- There exist renamings τ_1 and τ_2 such that $\ell_1\tau_1 \to r_1\tau_1, \ell_2\tau_2 \to r_2\tau_2 \in \mathcal{R}$.
- The equality $\mathsf{Var}(\ell_1 \to r_1) \cap \mathsf{Var}(\ell_2 \to r_2) = \varnothing$ holds.
- The position p is a function position of ℓ_2.
- If $p = \epsilon$, then $\ell_1 \to r_1$ and $\ell_2 \to r_2$ are not variants.
- The terms ℓ_1 and $\ell_2|_p$ are unifiable.

Each critical overlap $(\ell_1 \to r_1, p, \ell_2 \to r_2)$ gives its *critical pair* $(\ell_2\sigma)[r_1\sigma]_p \approx r_2\sigma$. The set of all critical pairs of $\mathcal{R}$ is denoted by $\mathsf{CP}(\mathcal{R})$.

The next well-known lemma is attributed to Huet [15].

Lemma 45 (the Critical Pair Lemma)**.** *A terminating TRS is confluent if and only if its all critical pairs are joinable.*

Example 46 (continued from Example 28)**.** The term rewrite system has only one critical pair $\mathsf{0} \approx \mathsf{0}$ from the overlap $(\mathsf{0} - y \to \mathsf{0}, \epsilon, x - \mathsf{0} \to x)$. The critical pair is illustrated by the peak:

$$\mathsf{0} \leftarrow \mathsf{0} - \mathsf{0} \to \mathsf{0}$$

The peak is trivially joinable, and thus $\mathcal{R}_{\div}$ is confluent, and moreover it is complete.

Now we are ready to give a sufficient condition for decidability of the word problem of an equational system.

Definition 47. A term rewrite system $\mathcal{R}$ is called a *complete presentation* for an equational system $\mathcal{E}$ if $\mathcal{R}$ is complete, and $\leftrightarrow_{\mathcal{R}}^*$ and $\leftrightarrow_{\mathcal{E}}^*$ coincide.

Proposition 48. *The word problem of an equational system $\mathcal{E}$ is decidable if there is a finite complete presentation $\mathcal{R}$ for $\mathcal{E}$.*

Proof. The following algorithm gives a decision procedure: let $s \approx t$ be an equation given as an input.

1. Compute normal forms of s and t, say $s{\downarrow}$ and $t{\downarrow}$.
2. Answer affirmatively if $s{\downarrow}$ and $t{\downarrow}$ are identical, or negatively otherwise.

Termination of the algorithm is guaranteed by the termination of $\mathcal{R}$. We show completeness: $s \leftrightarrow_{\mathcal{E}}^* t$ and $s \leftrightarrow_{\mathcal{R}}^* t$ are equivalent since $\mathcal{R}$ is a complete presentation for $\mathcal{E}$. By the Church–Rosser property, s and t eventually join at their normal form if and only if $s \leftrightarrow_{\mathcal{R}}^* t$. ☐

Example 49 (continued from Example 46)**.** The equational system $\mathcal{E}_{\div}$ has a complete presentation $\mathcal{R}_{\div}$. Now we can disprove the invalid equation $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0})) \approx \mathsf{s}(\mathsf{0})$ without finding a model. The left-hand side $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{0}))) \div \mathsf{s}(\mathsf{s}(\mathsf{0}))$ has a normal form $\mathsf{s}(\mathsf{s}(\mathsf{0}))$ with respect to $\rightarrow_{\mathcal{R}}$, but the right-hand side $\mathsf{s}(\mathsf{0})$ is already a normal form. Hence, the equation is invalid under $\mathcal{E}_{\div}$.

2.3 Ordered Rewriting

We introduce the notion of ordered rewriting [23].

Definition 50 (ordered rewriting system)**.** Let $\mathcal{E}$ be an equational system and $>$ a context-closed proper order on terms. The pair $\mathcal{O} = (\mathcal{E}, >)$ is called an *ordered term rewrite system* (*OTRS*). The relation $\rightarrow_{\mathcal{O}}$ is defined on terms as follows: $s \rightarrow_{\mathcal{O}} t$ if there exist an equation $\ell \approx r \in \mathcal{E} \cup \mathcal{E}^{-1}$, a context C, and a substitution σ such that $s = C[\ell\sigma]$, $t = C[r\sigma]$, and $\ell\sigma > r\sigma$.

Example 51. Let $\mathcal{E}$ be the equational system of the four equations:

$$\begin{array}{cc} \mathsf{0} + y \approx y & x + \mathsf{s}(y) \approx \mathsf{s}(x + y) \\ (x + y) + z \approx x + (y + z) & x + y \approx y + x \end{array}$$

and let $>_{\mathsf{lpo}}$ be the lexicographic path order induced by the precedence $+ > \mathsf{s} > \mathsf{0}$. The pair $\mathcal{O} = (\mathcal{E}, >_{\mathsf{lpo}})$ is an OTRS and admits a sequence $\mathsf{s}(\mathsf{0}) + \mathsf{0} \rightarrow_{\mathcal{O}} \mathsf{0} + \mathsf{s}(\mathsf{0}) \rightarrow_{\mathcal{O}} \mathsf{s}(\mathsf{0}) \in \mathsf{NF}(\mathcal{O})$. In contrast, $x + y \rightarrow_{\mathcal{E}} y + x$ is not a step by $\rightarrow_{\mathcal{O}}$ as $x + y >_{\mathsf{lpo}} y + x$ does not hold.

Definition 52. Let $\mathcal{O}$ be an OTRS and $\mathcal{E}$ an ES.

- The OTRS $\mathcal{O}$ is *ground-terminating* if the restriction of $\to_{\mathcal{O}}$ to ground terms is terminating.
- The OTRS $\mathcal{O}$ is *ground-confluent* if the restriction of $\to_{\mathcal{O}}$ to ground terms is confluent.
- The OTRS $\mathcal{O}$ is *ground-complete* if $\to_{\mathcal{O}}$ is ground-terminating and ground-confluent.
- The OTRS $\mathcal{O}$ is *ground-equivalent* to $\mathcal{E}$ if the restrictions of $\leftrightarrow^*_{\mathcal{O}}$ and $\leftrightarrow^*_{\mathcal{E}}$ coincide.
- The OTRS $\mathcal{O}$ is a *ground-complete presentation* for $\mathcal{E}$ if $\mathcal{O}$ is ground-complete and ground-equivalent to $\mathcal{E}$.
- A reduction order $>$ is *ground-total* if every two different ground terms s, t satisfy either $s > t$ or $t > s$.

The following proposition states that, given a ground equation $s \approx t$ and an equational system $\mathcal{E}$, validity of $s \approx t$ under $\mathcal{E}$ is decidable if $\mathcal{E}$ admits a ground-complete OTRS.

Proposition 53. *Let $\mathcal{O} = (\mathcal{E}', >)$ be a ground-complete presentation for $\mathcal{E}$, and let s, t be ground terms. Then $s \leftrightarrow^*_{\mathcal{E}} t$ if and only if $s{\downarrow_{\mathcal{O}}} = t{\downarrow_{\mathcal{O}}}$.*

How can ground completeness of OTRSs be verified? Usually OTRSs employ reduction orders and such OTRSs fulfil ground termination.

Proposition 54. *Every OTRS is terminating if its ordering is a reduction order.*

Ground confluence of terminating OTRSs is characterized by ground joinability of extended critical pairs.

Definition 55 (extended critical overlap/peak/pair)**.** Let $\mathcal{O} = (\mathcal{E}, >)$ be an OTRS, $\ell_1 \approx r_1$ and $\ell_2 \approx r_2$ equations, and $p \in \mathsf{Pos}(\ell_2)$. The triple $(\ell_1 \approx r_1, p, \ell_2 \approx r_2)$ is an *extended overlap* if all the following conditions are satisfied:

- There exist renamings τ_1 and τ_2 such that $\ell_1\tau_1 \approx r_1\tau_1, \ell_2\tau_2 \approx r_2\tau_2 \in \mathcal{E} \cup \mathcal{E}^{-1}$.
- The equality $\mathsf{Var}(\ell_1 \approx r_1) \cap \mathsf{Var}(\ell_2 \approx r_2) = \varnothing$ holds.

- The position p is not a function position of ℓ_2.
- If $p = \epsilon$ then $\ell_1 \approx r_1$ and $\ell_2 \approx r_2$ are not variants.
- The terms ℓ_1 and $\ell_2|_p$ has an mgu σ.
- Neither $r_1\sigma > \ell_1\sigma$ nor $r_2\sigma > \ell_2\sigma$ holds.

The set of all extended critical pairs is denoted by $\mathsf{ECP}(\mathcal{O})$. Each extended critical overlap $(\ell_1 \approx r_1, p, \ell_2 \approx r_2)$ gives *extended critical pair* $(\ell_2\sigma)[r_1\sigma]_p \approx r_2\sigma$ where σ is an mgu of ℓ_1 and $\ell_2|_p$.

Example 56 (continued from Example 51)**.** The OTRS admits the extended critical pair $\mathsf{s}(x + y) \approx \mathsf{s}(y) + x$, which originates from the extended overlap $(x + \mathsf{s}(y) \approx \mathsf{s}(x + y), \epsilon, x + y \approx y + x)$. The overlap is illustrated by the following conversion:

$$\mathsf{s}(x + y) \ {}_{\mathcal{E}}\!\leftarrow x + \mathsf{s}(y) \rightarrow_{\mathcal{E}} \mathsf{s}(y) + x$$

Definition 57. Given an OTRS $\mathcal{O}$, we say that s and t are *ground-joinable* if $s\sigma \downarrow_{\mathcal{O}} t\sigma$ for all ground substitutions σ for s and t, and we write $s \Downarrow_{\mathcal{O}} t$.

Lemma 58 (the Extended Critical Pair Lemma [23])**.** *Let $\mathcal{O}$ be an OTRS equipped with a ground-total reduction order. The OTRS $\mathcal{O}$ is ground-confluent if and only if* $\mathsf{ECP}(\mathcal{O}) \subseteq \Downarrow_{\mathcal{O}}$.

The Extended Critical Pair Lemma demands ground-totality.

Example 59. Let $\mathcal{E} = \{\mathsf{f}(x) \approx \mathsf{a}, \mathsf{b} \approx \mathsf{c}\}$, and let $>_{\mathsf{lpo}}$ be the LPO induced by the precedence $\mathsf{b} \succ \mathsf{a}, \mathsf{c}$. The OTRS $\mathcal{O} = (\mathcal{E}, >_{\mathsf{lpo}})$ have no extended critical pair, and thus the condition $\mathsf{ECP}(\mathcal{O}) \subseteq \Downarrow_{\mathcal{O}}$ is vacuously satisfied. However, $\mathcal{O}$ is not ground-confluent: $\mathsf{f}(\mathsf{c}) \ {}_{\mathcal{O}}\!\leftarrow \mathsf{f}(\mathsf{b}) \rightarrow_{\mathcal{O}} \mathsf{a}$ but $\mathsf{f}(\mathsf{c}), \mathsf{a} \in \mathsf{NF}(\mathcal{O})$ as a and c are not comparable with respect to $>_{\mathsf{lpo}}$.

A major problem when testing ground joinability is that there are infinitely many ground substitutions in general. Martin and Nipkow [23] suggest extending ordered rewriting by variable orders. We define an extension of LPO.

Definition 60 (extended lexicographic path orders)**.** Let $>$ be a precedence and $\succ$ a strict order on variables. We define the binary relation $>_{\mathsf{elpo}}$ on terms as the smallest relation satisfying the following conditions:

- $f(s_1, \ldots, s_n) >_{\mathsf{elpo}} f(t_1, \ldots, t_n)$ if there exists i such that
  - $s_1 = t_1, \ldots, s_{i-1} = t_{i-1}$,

  - $s_i >_{\mathsf{elpo}} t_i$, and
  - $s >_{\mathsf{elpo}} t_{i+1}, \ldots, s >_{\mathsf{elpo}} t_n$.

- $s = f(s_1, \ldots, s_n) >_{\mathsf{elpo}} g(t_1, \ldots, t_m)$ if $f > g$ and $s >_{\mathsf{elpo}} t_i$ for all i.
- $f(s_1, \ldots, s_n) >_{\mathsf{elpo}} t$ if there exists i such that $s_i >^{=}_{\mathsf{elpo}} t$.
- $x >_{\mathsf{elpo}} y$ if $x \succ y$.

When $\succ = \varnothing$, the extended LPO coincides with the standard LPO.

Theorem 61 ([23])**.** *Let $\mathcal{O} = (\mathcal{E}, >_{\mathsf{lpo}})$ be an OTRS. The relation $s \Downarrow_{\mathcal{O}} t$ holds if $s\rho \downarrow_{(\mathcal{E}, >_{\mathsf{elpo}})} t\rho$ for all variable substitutions ρ and total orders $\succ$ on variables, where $>_{\mathsf{elpo}}$ is the extended LPO induced by $>$ and $\succ$.*

Example 62 (associativity and commutativity)**.** Consider the OTRS $\mathcal{O}$ consisting of the equational system $\mathcal{E}$ over the signature $\{*^{(2)}, a^{(0)}, b^{(0)}\}$:

$$\begin{array}{lr}
(1) & (x * y) * z \approx x * (y * z) \\
(2) & x * y \approx y * x \\
(3) & x * (y * z) \approx y * (x * z)
\end{array}$$

and the LPO with the total precedence $* > a > b$. Note that (1) is oriented from left to right by LPO $>_{\mathsf{lpo}}$ and (2) (3) are just a renaming. Therefore, we can omit some cases to calculate extended critical pairs. For example,

- We obtain $(x * y) * (z * w) \approx (x * (y * z)) * w$ by embedding (1) into itself.
- We obtain $z * (x * y) \approx x * (y * z)$ by embedding (1) into (2).
- We obtain $y * (x * z) \approx x * (z * y)$ and $y * (x * z) \approx (y * z) * x$ by embedding (2) into (3). Note that there are two ways to embed (2) into (3).

The ordered rewriting system admits 8 distinct extended critical pairs:

$$\begin{array}{cc}
x * (y * z) \approx (y * x) * z & x * ((y * z) * w) \approx (y * (x * z)) * w \\
(x * y) * (z * w) \approx (x * (y * z)) * w & z * (x * y) \approx x * (y * z) \\
(y * z) * x \approx y * (x * z) & (y * z) * (x * w) \approx x * (y * (z * w)) \\
y * (x * (z * w)) \approx x * (z * (y * w)) & y * (x * z) \approx (y * z) * x
\end{array}$$

and all of them are ground joinable. For example, $x * (y * z)$ and $(y * x) * z$ are ground joinable: $(y * x) * z \longrightarrow_{\mathcal{O}} y * (x * z)$ and

- if $x \succ y$ then $x * (y * z) \longrightarrow_{\mathcal{E},>_{\mathsf{elpo}}} y * (x * z)$.
- if $x = y$ then the terms $x * (y * z)$ and $y * (x * z)$ are identical.
- if $y \succ x$ then $y * (x * z) \longrightarrow_{\mathcal{E},>_{\mathsf{elpo}}} x * (y * z)$.

Here, for simplicity, a variable x denotes $x\rho$.

Chapter 3

Generalized Weighted Path Order

First we introduce the weighted path order [34] and prove that the weighted path order is a monotonic semantic path order [11]. Based on this observation we give a generalization of the weighted path order, dubbed the generalized weighted path order (GWPO). In addition, we give several ways to instantiate the GWPO, which is useful to construct non-simplification orders. Next we identify the relationships between the GWPO and well-known classes of reduction orders, such as the Knuth–Bendix order and the lexicographic path order. For the rest of this chapter we discuss implementations of the GWPO and experiments.

3.1 The Weighted Path Order

We introduce the weighted path order [34] and prove that the weighted path order is a monotonic semantic path order [11].

Definition 63 (the weighted path order)**.** Let $\mathcal{A}$ be a well-founded algebra and $\succ$ a precedence. The *weighted path order (WPO)* $>_{\mathsf{wpo}}$ is defined on terms as follows: $s >_{\mathsf{wpo}} t$ if

1. $s >_{\mathcal{A}} t$, or
2. $s \geq_{\mathcal{A}} t$, $s = f(s_1, \ldots, s_m)$, and one of the following conditions holds.
   (a) $s_i >^{=}_{\mathsf{wpo}} t$ for some $1 \leq i \leq m$.
   (b) $t = g(t_1, \ldots, t_n)$ and $s >_{\mathsf{wpo}} t_j$ for all $1 \leq j \leq n$, and moreover
   - $f \succ g$, or

- $f = g$, and $(s_1, \ldots, s_m) >^{\mathsf{lex}}_{\mathsf{wpo}} (t_1, \ldots, t_n)$.

Theorem 64 ([34])**.** *Every weighted path order is a simplification order if its algebra is simple monotone.*

We show that the weighted path order characterizes the simplification order.

Lemma 65. *If $s >_{\mathsf{wpo}} t$ then $s \geq_{\mathcal{A}} t$.*

Proof. The claim follows immediately from the definition. □

Proposition 66. *Given a simplification order $>$, there is a simple monotone algebra and a precedence such that the induced weighted path order coincides with $>$.*

Proof. Given a simplification order $>$ consider$>_{\mathsf{wpo}}$ induced the term algebra $\mathcal{A}$ equipped with the ordering $>$, and the empty precedence. Obviously $>$ is simple monotone as it is a simplification order. If $s > t$, then immediately $s >_{\mathsf{wpo}} t$ as $s >_{\mathcal{A}} t$. Conversely, suppose $s >_{\mathsf{wpo}} t$. With an appeal to Lemma 65 we obtain $s \geq_{\mathcal{A}} t$, which yields $s > t$ or $s = t$ in this setting. The latter is impossible as $s >_{\mathsf{wpo}} t$. □

Next we introduce the monotonic semantic path order.

Definition 67. A pair $(\gtrsim, \sqsupseteq)$ is a *quasi-rewrite pair* if

- $\gtrsim$ is a rewrite preorder,
- $\sqsupseteq$ is a quasi-order on non-variable terms that is closed under substitutions, and
- $(\gtrsim, \sqsupseteq)$ has the *harmony property*:

  $$s_i \gtrsim t \Longrightarrow f(s_1, \ldots, s_i, \ldots, s_n) \sqsupseteq f(s_1, \ldots, t, \ldots, s_n)$$

  for all $n \in \mathbb{N}$, n-ary function symbols f, terms $s_1, \ldots, s_n, t$ and argument positions i.

A quasi-rewrite pair $(\gtrsim, \sqsupseteq)$ is a *quasi-reduction pair* if the strict part $\sqsupset$ of $\sqsupseteq$ is well-founded and closed under substitutions.

Definition 68 (the monotonic semantic path order)**.** Let $(\gtrsim, \sqsupseteq)$ be a quasi-reduction pair. The *semantic path order* $>_{\mathsf{spo}}$ is defined on terms as follows: $s >_{\mathsf{spo}} t$ if $s = f(s_1, \ldots, s_m)$ and one of the followings holds.

1. $s_i >^{=}_{\mathsf{spo}} t$ for some $1 \leq i \leq m$.
2. $t = g(t_1, \ldots, t_n)$ and $s >_{\mathsf{spo}} t_j$ for all $1 \leq j \leq n$, and moreover
   - $s \sqsupset t$, or
   - $s \underset{\sim}{\sqsupset} t$, $f = g$, and $(s_1, \ldots, s_m) >^{\mathsf{lex}}_{\mathsf{spo}} (t_1, \ldots, t_n)$.

We define the *monotonic semantic path order* $>_{\mathsf{mspo}}$ as follows: $s >_{\mathsf{mspo}} t$ if $s \gtrsim t$ and $s \sqsupset t$.

Proposition 69 ([17])**.** *Every semantic path order is a well-founded order closed under substitutions.*

Theorem 70 ([11])**.** *Every monotonic semantic path order is a reduction order.*

We prove that every weighted path order is a monotonic semantic path order. Let $>_{\mathsf{wpo}}$ be a weighted path order induced by a weakly monotone algebra $\mathcal{A}$ and a precedence $\succ$. Define the relation $\underset{\sim}{\sqsupset}$ on non-variable terms as follows: $s \underset{\sim}{\sqsupset} t$ if $s >_{\mathcal{A}} t$, or $s \geq_{\mathcal{A}} t$ and $\mathsf{root}(s) \succ^{=} \mathsf{root}(t)$. First we analyze the strict part $\sqsupset$ of $\underset{\sim}{\sqsupset}$.

Lemma 71. *The relation $s \sqsupset t$ holds if and only if $s >_{\mathcal{A}} t$, or $s \geq_{\mathcal{A}} t$ and $\mathsf{root}(s) \succ \mathsf{root}(t)$.*

Lemma 72. *The pair $(\gtrsim, \underset{\sim}{\sqsupset})$ is a quasi-reduction pair.*

Let $>_{\mathsf{spo}}$ be the semantic path order induced by $(\gtrsim, \underset{\sim}{\sqsupset})$.

Lemma 73. *Suppose $\mathcal{A}$ is simple. If $s >_{\mathsf{wpo}} t$, then $s >_{\mathsf{spo}} t$.*

Proof. We show the claim by induction on the sum $|s| + |t|$. Let $s = f(s_1, \ldots, s_m)$. We analyze the derivation of $s >_{\mathsf{wpo}} t$, and distinguish the four cases.

- Suppose that $s >_{\mathsf{wpo}} t$ is derived from $s >_{\mathcal{A}} t$. The case when t is a variable is trivial. Let $t = g(t_1, \ldots, t_n)$. From $s >_{\mathcal{A}} t$ we have $s \sqsupset t$. Since $\mathcal{A}$ is simple monotone, for all $j \in \{1, \ldots, n\}$ we have $t \geq_{\mathcal{A}} t_j$. Thus, for all $j \in \{1, \ldots, n\}$ we have $s >_{\mathcal{A}} t_j$ and moreover $s >_{\mathsf{wpo}} t_j$. Now we have the following derivation of $s >_{\mathsf{spo}} t$:

$$\dfrac{s \sqsupset t \qquad \dfrac{\forall j (s >_{\mathsf{wpo}} t_j)}{\forall j (s >_{\mathsf{spo}} t_j)}\,\text{I.H.}}{s >_{\mathsf{spo}} g(t_1, \ldots, t_n) = t}$$

- Suppose $s >_{\mathsf{wpo}} t$ is derived from $s \geq_{\mathcal{A}} t$ and $s_i >_{\mathsf{wpo}}^{=} t$ for some $i \in \{1, \ldots, m\}$. By the induction hypothesis we have $s_i >_{\mathsf{spo}}^{=} t$, and thus $s >_{\mathsf{spo}} t$.

- Suppose that $s >_{\mathsf{wpo}} t$ is derived as follows:

$$\frac{s \geq_{\mathcal{A}} t \qquad f \succ g \qquad \forall j (s >_{\mathsf{wpo}} t_j)}{s = f(s_1, \ldots, s_m) >_{\mathsf{wpo}} g(t_1, \ldots, t_n) = t}$$

From $s \geq_{\mathcal{A}} t$ and $f \succ g$ we have $s \sqsupset t$. Thus, we have the following derivation of $s >_{\mathsf{spo}} t$:

$$\frac{s \sqsupset t \qquad \dfrac{\forall j (s >_{\mathsf{wpo}} t_j)}{\forall j (s >_{\mathsf{spo}} t_j)}\,\text{I.H.}}{s >_{\mathsf{spo}} g(t_1, \ldots, t_n) = t}$$

- Suppose that $s >_{\mathsf{wpo}} t$ is derived as follows:

$$\frac{s \geq_{\mathcal{A}} t \qquad \forall j (s >_{\mathsf{wpo}} t_j) \qquad (s_1, \ldots, s_m) >_{\mathsf{wpo}}^{\mathsf{lex}} (t_1, \ldots, t_m)}{s = f(s_1, \ldots, s_m) >_{\mathsf{wpo}} f(t_1, \ldots, t_m) = t}$$

From $s \geq_{\mathcal{A}} t$ and that s and t have the same root symbol, we have $s \sqsupseteq\!\!\!\sim t$. Thus, we have the following derivation of $s >_{\mathsf{spo}} t$:

$$\frac{s \sqsupseteq\!\!\!\sim t \qquad \dfrac{\forall j (s >_{\mathsf{wpo}} t_j)}{\forall j (s >_{\mathsf{spo}} t_j)}\,\text{I.H.} \qquad \dfrac{(s_1, \ldots, s_m) >_{\mathsf{wpo}}^{\mathsf{lex}} (t_1, \ldots, t_m)}{(s_1, \ldots, s_m) >_{\mathsf{spo}}^{\mathsf{lex}} (t_1, \ldots, t_m)}\,\text{I.H.}}{s = f(s_1, \ldots, s_m) >_{\mathsf{spo}} f(t_1, \ldots, t_m) = t}$$

This case concludes the proof. □

Lemma 74. *Suppose $\mathcal{A}$ is simple. If $s >_{\mathsf{spo}} t$, then $s >_{\mathsf{wpo}} t$.*

Proof. We show the claim by induction on the sum $|s| + |t|$. Let $s = f(s_1, \ldots, s_m)$. We analyze the derivation of $s >_{\mathsf{spo}} t$, and distinguish the three cases.

- If $s_i >_{\mathsf{spo}}^{=} t$ for some $i \in \{1, \ldots, m\}$, by the induction hypothesis we have $s_i >_{\mathsf{wpo}}^{=} t$. From the simplicity and Lemma 65 we have $s \geq_{\mathcal{A}} s_i \geq_{\mathcal{A}} t_i$ and thus $s >_{\mathsf{wpo}} t$.

- Suppose that $s >_{\mathsf{spo}} t$ is derived as follows:

$$\frac{s \sqsupset t \qquad \forall j(s >_{\mathsf{spo}} t_j)}{s = f(s_1, \ldots, s_m) >_{\mathsf{spo}} g(t_1, \ldots, t_n) = t}$$

We analyze $s \sqsupset t$ and further distinguish two cases:

– If $s >_{\mathcal{A}} t$ then immediately $s >_{\mathsf{wpo}} t$.

– Otherwise, we have $s \geq_{\mathcal{A}} t$ and $f \succ g$. In this case we have the following derivation:

$$\frac{s \geq_{\mathcal{A}} t \qquad f \succ g \qquad \dfrac{\forall j(s >_{\mathsf{spo}} t_j)}{\forall j(s >_{\mathsf{wpo}} t_j)}\ \text{I.H.}}{s = f(s_1, \ldots, s_m) >_{\mathsf{wpo}} g(t_1, \ldots, t_n) = t}$$

This concludes the analysis of the subcases.

- Suppose that $s >_{\mathsf{spo}} t$ is derived as follows:

$$\frac{s \sqsupset\!\!\!\sim t \qquad \forall j(s >_{\mathsf{spo}} t_j) \qquad (s_1, \ldots, s_m) >^{\mathsf{lex}}_{\mathsf{spo}} (t_1, \ldots, t_m)}{s = f(s_1, \ldots, s_m) >_{\mathsf{spo}} f(t_1, \ldots, t_m) = t}$$

From $s \sqsupset\!\!\!\sim t$ we have $s \geq_{\mathcal{A}} t$. Thus, we have the following derivation:

$$\frac{s \geq_{\mathcal{A}} t \qquad \dfrac{\forall j(s >_{\mathsf{spo}} t_j)}{\forall j(s >_{\mathsf{wpo}} t_j)}\ \text{I.H.} \qquad \dfrac{(s_1, \ldots, s_m) >^{\mathsf{lex}}_{\mathsf{spo}} (t_1, \ldots, t_m)}{(s_1, \ldots, s_m) >^{\mathsf{lex}}_{\mathsf{wpo}} (t_1, \ldots, t_m)}\ \text{I.H.}}{s = f(s_1, \ldots, s_m) >_{\mathsf{wpo}} f(t_1, \ldots, t_m) = t}$$

This case concludes the proof. □

Theorem 75. *Let $>_{\mathsf{wpo}}$ be a weighted path order induced by a simple monotone well-founded algebra and a precedence. There is a quasi-reduction pair such that the induced monotonic semantic path order coincides with $>_{\mathsf{wpo}}$.*

3.2 A Generalization of the WPO

Observe that Theorem 75 requires a simple monotone algebra. However, the construction of the reduction pair is still valid even if the algebra is no longer simple. Based on this observation, we give an alternative definition of the weighted path order, which coincides with the original weighted path order when its algebra is simple. Moreover, by the knowledge of the monotonic semantic path order, we obtain a generalization of the weighted path order.

Definition 76 (the generalized weighted path order)**.** Let $\mathcal{A}$ be a weakly-monotone well-founded algebra and $\succ$ a precedence. We define the binary relation $>_{\mathsf{wpo}'}$ on terms as the smallest relation satisfying the following conditions:

(subterm) $f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} t$ if $s_i >^{=}_{\mathsf{wpo}'} t$ for some term s_i.

(algebra) $s >_{\mathsf{wpo}'} g(t_1, \ldots, t_m)$ if

- $s >_{\mathcal{A}} g(t_1, \ldots, t_m)$, and
- $s >_{\mathsf{wpo}'} t_j$ for all $j \in \{1, \ldots, m\}$.

(lex) $f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} f(t_1, \ldots, t_n)$ if

- $f(s_1, \ldots, s_n) \geq_{\mathcal{A}} f(t_1, \ldots, t_n)$,
- $(s_1, \ldots, s_n) >^{\mathsf{lex}}_{\mathsf{wpo}'} (t_1, \ldots, t_n)$, and
- $f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} t_j$ for all $j \in \{1, \ldots, n\}$.

(precedence) $f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} g(t_1, \ldots, t_m)$ if

- $f(s_1, \ldots, s_n) \geq_{\mathcal{A}} g(t_1, \ldots, t_m)$,
- $f \succ g$, and
- $f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} t_j$ for all $j \in \{1, \ldots, m\}$.

Informally $>_{\mathsf{wpo}'}$ is given by the following inference system:

$$\frac{\exists i(s_i >^{=}_{\mathsf{wpo}'} t)}{f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} t}\ \text{(subterm)}$$

$$\frac{s >_{\mathcal{A}} t \qquad \forall j(s >_{\mathsf{wpo}'} t_j)}{s >_{\mathsf{wpo}'} g(t_1, \ldots, t_n) = t}\ \text{(algebra)}$$

$$\frac{s \geq_{\mathcal{A}} t \qquad (s_1, \ldots, s_n) >^{\mathsf{lex}}_{\mathsf{wpo}'} (t_1, \ldots, t_n) \qquad \forall j(s >_{\mathsf{wpo}'} t_j)}{s = f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} f(t_1, \ldots, t_n) = t}\ \text{(lex)}$$

$$\frac{s \geq_{\mathcal{A}} t \qquad f \succ g \qquad \forall j(s >_{\mathsf{wpo}'} t_j)}{s = f(s_1, \ldots, s_n) >_{\mathsf{wpo}'} g(t_1, \ldots, t_m) = t}\ \text{(precedence)}$$

In particular, the inference system is convenient when we give a proof of $s >_{\mathsf{wpo}'} t$ for given terms s and t. The *generalized weighted path order* $>_{\mathsf{gwpo}}$ is defined as follows: $s >_{\mathsf{gwpo}} t$ if $s \geq_{\mathcal{A}} t$ and $s >_{\mathsf{wpo}'} t$.

Theorem 77. *For every simple monotone algebra and precedence, the induced relations $>_{\mathsf{wpo}'}$, $>_{\mathsf{gwpo}}$ and $>_{\mathsf{wpo}}$ coincide.*

Proof. Construct the quasi-reduction pair $(\geq_{\mathcal{A}}, \sqsupset\!\!\!\sim)$ in the same manner as the proof of Theorem 75. From Lemmata 73 and 74 the semantic path order $>_{\mathsf{spo}}$ induced by the reduction pair coincides with $>_{\mathsf{wpo}}$. It is easy to verify that $>_{\mathsf{spo}}$ and $>_{\mathsf{wpo}'}$ coincide by the definitions. With an appeal to Lemma 65 and Theorem 75 it is easy to verify that $>_{\mathsf{wpo}}$ and $>_{\mathsf{gwpo}}$ coincide. ☐

Theorem 78. *For every weakly-monotone well-founded algebra and precedence, the induced generalized weighted path order is a reduction order.*

Proof. Construct the quasi-reduction pair $(\geq_{\mathcal{A}}, \sqsupset\!\!\!\sim)$ in the same manner as the proof of Theorem 75. It is easy to verify that the induced monotonic semantic path order $>_{\mathsf{mspo}}$ coincides with $>_{\mathsf{gwpo}}$. ☐

The following example show that the orientation by $\geq_{\mathcal{A}}$ in the definition of $>_{\mathsf{gwpo}}$ cannot be dropped. In other words, it shows that $>_{\mathsf{wpo}'}$ is not a reduction order by itself if its algebra is not simple.

Example 79. Consider the non-terminating TRS $\mathcal{R}$ taken from [32]:

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

Take the empty precedence and the following weakly-monotone well-founded algebra:

- carrier: $\{0,1\} \times \mathbb{N}$
- order $>$ on carrier: $(x,y) > (z,w)$ if $x = z$ and $y >_{\mathbb{N}} w$.
- interpretation: $\mathsf{a}_{\mathcal{A}} = (0,1)$, and $\mathsf{b}_{\mathcal{A}} = (1,1)$, and $\mathsf{g}_{\mathcal{A}}(x,y) = (0,0)$, and

$$\mathsf{f}_{\mathcal{A}}((x_1,y_1),(x_2,y_2),(x_3,y_3)) = \begin{cases} (0, y_1 + y_2 + y_3) & (x_1 = x_2) \\ (0, y_1 + y_2 + 3y_3) & (x_1 \neq x_2) \end{cases}$$

Note that $\mathcal{A}$ is not simple because of $\mathsf{g}_{\mathcal{A}}(x,y) = (0,0)$. It is readily verified that $\mathsf{f}(\mathsf{a},\mathsf{b},x) >_{\mathcal{A}} \mathsf{f}(x,x,x)$ by computation: we have

$$\mathsf{f}_{\mathcal{A}}(\mathsf{a}_{\mathcal{A}}, \mathsf{b}_{\mathcal{A}}, (n,m)) = (0, 3m+2) > (0, 3m) = \mathsf{f}_{\mathcal{A}}((n,m),(n,m),(n,m))$$

for all $n \in \{0,1\}$ and $m \in \mathbb{N}$. Hence $\mathsf{f}(\mathsf{a},\mathsf{b},x) >_{\mathsf{wpo}'} \mathsf{f}(x,x,x)$:

$$\dfrac{\mathsf{f}(\mathsf{a},\mathsf{b},x) >_{\mathcal{A}} \mathsf{f}(x,x,x) \qquad \dfrac{x = x}{\mathsf{f}(\mathsf{a},\mathsf{b},x) >_{\mathsf{wpo}'} x} \qquad \cdots}{\mathsf{f}(\mathsf{a},\mathsf{b},x) >_{\mathsf{wpo}'} \mathsf{f}(x,x,x)}$$

By using the subterm rule, we have $\mathsf{g}(x,y) >_{\mathsf{wpo}'} x$ and $\mathsf{g}(x,y) >_{\mathsf{wpo}'} y$. Thus, the system has the compatibility with $>_{\mathsf{wpo}'}$, but it is not terminating:

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

Therefore, compatibility with $>_{\mathsf{wpo}'}$ does not imply termination if the algebra of $>_{\mathsf{wpo}'}$ is not simple. Note that, in this case, $\mathsf{g}(x,y) \geq_{\mathcal{A}} x$ and $\mathsf{g}(x,y) \geq_{\mathcal{A}} y$ are not satisfied.

For the rest of this section, we introduce three subclasses of the GWPO with polynomial interpretation over $\mathbb{N}$.

Definition 80. The subclass $\mathsf{gwpo}_{\mathbb{N}}$ is the subclass of the GWPO that uses polynomial interpretation over $\mathbb{N}$ as its algebra. Formally, its algebra is given as follows:

> carrier: the set of all natural numbers $\mathbb{N}$ equipped with the standard ordering on $\mathbb{N}$.
>
> interpretation: linear polynomials in the form $f_{\mathcal{A}}(x_1,\ldots,x_n) = a_0 + a_1x_1 + \ldots + a_nx_n$ where f is an n-ary function symbol and $a_0,\ldots,a_n \in \mathbb{N}$.

The subclass $\mathsf{wpo}_{\mathbb{N}}$ is the subclass of $\mathsf{gwpo}_{\mathbb{N}}$ that only uses positive natural numbers for its coefficients. Formally its algebra is given as follows:

> carrier: the set of all natural numbers $\mathbb{N}$ equipped with the standard ordering on $\mathbb{N}$.
>
> interpretation: linear polynomials in the form $f_{\mathcal{A}}(x_1,\ldots,x_n) = a_0 + a_1x_1 + \cdots + a_nx_n$ where f is an n-ary function symbol, $a_1,\ldots,a_n$ are positive natural numbers, and $a_0 \in \mathbb{N}$.

The subclass gwpo_{01} is the subclass of $\mathsf{gwpo}_{\mathbb{N}}$ that only uses $\{0,1\}$ for its coefficients. Formally its algebra is given as follows:

> carrier: the set of all natural numbers $\mathbb{N}$ equipped with the standard ordering on $\mathbb{N}$.

interpretation: linear polynomials in the form $f_{\mathcal{A}}(x_1, \ldots, x_n) = a_0 + a_1x_1 + \cdots + a_nx_n$ where f is an n-ary function symbol, $a_1, \ldots, a_n \in \{0, 1\}$, and $a_0 \in \mathbb{N}$.

Search space of parameters for gwpo_{01} is restricted so that its implementation works efficiently. We discuss the efficiency in Section 3.4 and Section 3.5.

Example 81. Now we are ready to prove termination of the TRS of round-up division, given in Example 28. The rewrite rules are given as follows:

$$\begin{array}{rcl} \mathsf{0} - y \to \mathsf{0} & \qquad & \mathsf{0} \div \mathsf{s}(y) \to \mathsf{0} \\ x - \mathsf{0} \to x & & \mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y)) \\ \mathsf{s}(x) - \mathsf{s}(y) \to x - y & & \end{array}$$

Recall that the TRS is not simply terminating as shown in Example 41, but we can show termination via the GWPO, in particular via gwpo_{01}. Consider the following algebra $\mathcal{A}$ on $\mathbb{N}$ with precedence $\div \succ \mathsf{s}$:

$$\mathsf{0}_{\mathcal{A}} = 0 \qquad \mathsf{s}_{\mathcal{A}}(x) = x + 1 \qquad x -_{\mathcal{A}} y = x \qquad x \div_{\mathcal{A}} y = x$$

Note that this algebra $\mathcal{A}$ is not simple but weakly-monotone. First, we verify $\ell \geq_{\mathcal{A}} r$ for each rule $\ell \to r$. Each rule is interpreted as follows:

$$\begin{array}{rcl} 0 \geq 0 & \qquad & 0 \geq 0 \\ x \geq x & & x + 1 \geq x + 1 \\ x + 1 \geq x & & \end{array}$$

This shows the orientation by $\geq_{\mathcal{A}}$. Next we verify $\ell >_{\mathsf{wpo}'} r$ for each rule $\ell \to r$. It is easy to verify $\mathsf{0} - y >_{\mathsf{wpo}'} \mathsf{0}$, and $x - \mathsf{0} >_{\mathsf{wpo}'} x$, and $\mathsf{0} \div \mathsf{s}(y) >_{\mathsf{wpo}'} \mathsf{0}$ using the rule (subterm). We can derive $\mathsf{s}(x) - \mathsf{s}(y) >_{\mathsf{wpo}'} x - y$ as follows:

$$\dfrac{x + 1 > x \qquad \dfrac{\vdots}{\mathsf{s}(x) - \mathsf{s}(y) >_{\mathsf{wpo}'} x} \qquad \dfrac{\vdots}{\mathsf{s}(x) - \mathsf{s}(y) >_{\mathsf{wpo}'} y}}{\mathsf{s}(x) - \mathsf{s}(y) >_{\mathsf{wpo}'} x - y}$$

The omitted parts are easily done by the rule (subterm). To handle the final rule $\mathsf{s}(x) \div \mathsf{s}(y) \to \mathsf{s}((x - y) \div \mathsf{s}(y))$, we show $\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo}'} (x - y) \div \mathsf{s}(y)$:

$$\dfrac{x + 1 > x \qquad \dfrac{x + 1 > x \qquad \cdots}{\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo}'} x - y} \qquad \dfrac{\mathsf{s}(y) = \mathsf{s}(y)}{\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo}'} \mathsf{s}(y)}}{\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo}'} (x - y) \div \mathsf{s}(y)}$$

Now $\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo}'} \mathsf{s}((x - y) \div \mathsf{s}(y))$ is easily done:

$$\dfrac{x+1 \geq x+1 \qquad \div \succ \mathsf{s} \qquad \dfrac{\vdots}{\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo'}} (x-y) \div \mathsf{s}(y)}}{\mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{wpo'}} \mathsf{s}((x-y) \div \mathsf{s}(y))}$$

Thus we have the compatibility with $>_{\mathsf{gwpo}}$, which yields the termination of the system.

An instance of the GWPO is not necessarily ground-total, even if its precedence is total and its algebra is equipped with a total order.

Example 82 (continued from Example 81)**.** We show that the terms $\mathsf{s}(\mathsf{0})$ and $\mathsf{0} - \mathsf{s}(\mathsf{0})$ are not comparable with respect to $>_{\mathsf{gwpo}}$. We have $\mathsf{s}(\mathsf{0}) >_{\mathcal{A}} \mathsf{0} - \mathsf{s}(\mathsf{0})$, and thus $\mathsf{0} - \mathsf{s}(\mathsf{0}) >_{\mathsf{gwpo}} \mathsf{s}(\mathsf{0})$ is impossible. However, we have $\mathsf{0} - \mathsf{s}(\mathsf{0}) >_{\mathsf{wpo'}} \mathsf{s}(\mathsf{0})$ using the rule (subterm). Hence, $\mathsf{s}(\mathsf{0}) >_{\mathsf{gwpo}} \mathsf{0} - \mathsf{s}(\mathsf{0})$ is also impossible, as $>_{\mathsf{wpo'}}$ is a semantic path order, and thus a proper order.

By construction, $\mathsf{gwpo}_{\mathbb{N}}$ subsumes $\mathsf{wpo}_{\mathbb{N}}$ and gwpo_{01}, but gwpo_{01} and $\mathsf{wpo}_{\mathbb{N}}$ are not comparable as shown in Example 81 and the subsequent examples:

Example 83. The termination of SK90_2.04 in TPDB 11.0 is shown by $\mathsf{wpo}_{\mathbb{N}}$:

$$\mathsf{f}(x + \mathsf{0}) \to \mathsf{f}(x) \qquad\qquad x + (y + z) \to (x + y) + z$$

For instance, take the following simple monotone algebra $\mathcal{A}$:

$$\mathsf{f}_{\mathcal{A}}(x) = x \qquad\qquad \mathsf{0}_{\mathcal{A}} = 1 \qquad\qquad x +_{\mathcal{A}} y = x + 2y + 1$$

It is easy to verify $\ell >_{\mathsf{gwpo}} r$ for each rule $\ell \to r$. In contrast, the TRS is not compatible with gwpo_{01}. This is because of the associativity rule $x + (y + z) \to (x + y) + z$. We cannot apply the (lex) rule of the GWPO to this since $x >_{\mathsf{wpo'}} x + y$ does not hold. The only choice is the (algebra) rule, which requires $x + (y + z) >_{\mathcal{A}} (x + y) + z$. This is only possible by using a polynomial with coefficients greater than 1.

Example 84. The termination of AProVE_04_rta2 in TPDB 11.0 cannot be shown by gwpo_{01}.

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

Let $f_{\mathcal{A}}(x, y) = ax + by + c$ and $s_{\mathcal{A}}(x) = dx + e$. To orient $\mathsf{f}(x, \mathsf{s}(y)) \to \mathsf{f}(y, x)$, a variant of commutation, the only choice is the (algebra) rule, which requires $\mathsf{f}(x, \mathsf{s}(y)) >_{\mathcal{A}} \mathsf{f}(y, x)$. Thus, we obtain the constraints:

$$\begin{aligned} a &\geq b & \qquad bd &\geq a \\ be + c &> c \end{aligned}$$

By solving these constraints where $a, b, d, e \in \{0, 1\}$, we obtain $a = b = d = e = 1$. Under this condition, $\mathsf{f}(\mathsf{s}(x), y) \geq_{\mathcal{A}} \mathsf{f}(x, \mathsf{s}(x))$ is impossible since $\mathsf{f}_{\mathcal{A}}(\mathsf{s}_{\mathcal{A}}(x), y) = x + y + (c + 1)$ and $\mathsf{f}_{\mathcal{A}}(x, \mathsf{s}_{\mathcal{A}}(x)) = 2x + (c + 1)$. In contrast, the termination is shown by $\mathsf{wpo}_{\mathbb{N}}$. For instance, take the empty precedence and the following algebra on $\mathbb{N}$:

$$\mathsf{f}_{\mathcal{A}}(x, y) = 3x + 2y + 1 \qquad\qquad \mathsf{s}_{\mathcal{A}}(x) = 4x + 4$$

The compatibility with $>_{\mathsf{wpo}}$ is easily verified, as we have $\mathsf{f}(\mathsf{s}(x), y) >_{\mathcal{A}} \mathsf{f}(x, \mathsf{s}(x))$ and $\mathsf{f}(x, \mathsf{s}(y)) >_{\mathcal{A}} \mathsf{f}(y, x)$.

3.3 Simulation

Yamada, Kusakari and Sakabe [34] showed that the weighted path order subsumes the Knuth–Bendix order and the lexicographic path order. The fact that the generalized weighted path order subsumes the weighted path order yields the following result:

Corollary 85. *The generalized weighted path order subsumes the Knuth–Bendix order and the lexicographic path order.*

Proof. This is an immediate consequence of Theorem 77. □

From now on, we further analyze relationships between $\mathsf{gwpo}_{\mathbb{N}}$, $\mathsf{wpo}_{\mathbb{N}}$, gwpo_{01}, the KBO, and the LPO by following the construction due to [34].

The Knuth–Bendix Order. Both $\mathsf{wpo}_{\mathbb{N}}$ and gwpo_{01} subsume the KBO:

Proposition 86. *For every weight and precedence for a Knuth–Bendix order $>_{\mathsf{kbo}}$, there are a simple monotone well-founded algebra and a precedence such that the induced generalized weighted path order $>_{\mathsf{gwpo}}$ satisfies the following conditions:*

- $>_{\mathsf{gwpo}}$ *is an instance of* $\mathsf{wpo}_{\mathbb{N}}$ *and* gwpo_{01}
- $>_{\mathsf{kbo}} \subseteq >_{\mathsf{gwpo}}$

Proof. Let (w, w_0) be a weight, $\succ$ a precedence for $>_{\mathsf{kbo}}$. We follow the construction of [34]; for each n-ary function symbol f, define its interpretation $f_{\mathcal{A}}$ as follows:

$$f_{\mathcal{A}}(x_1, \ldots, x_n) = w(f) - w_0 + \sum_{k=1}^{n} (x_k + w_0)$$

The generalized weighted path order $>_{\mathsf{gwpo}}$ induced by $\mathcal{A}$ and $\succ$ satisfies the conditions. □

Example 87. Consider the following term rewrite system [19]:

$$\begin{array}{rrr}
x + \mathsf{0} \to x & x + (-x) \to \mathsf{0} & (x + y) + z \to x + (y + z) \\
\mathsf{0} + x \to x & (-x) + x \to \mathsf{0} & -(x + y) \to (-x) + (-y) \\
-(-x) \to x & x + ((-x) + y) \to y & \\
-\mathsf{0} \to \mathsf{0} & (-x) + (x + y) \to y &
\end{array}$$

The term rewrite system is compatible with the Knuth–Bendix order induced by the weight $w(\mathsf{0}) = w(+) = w_0 = 1$ and $w(-) = 0$, and the precedence $- \succ + \succ \mathsf{0}$. For instance, $-(x + y) >_{\mathsf{kbo}} (-x) + (-y)$ is derived from $w(-(x+y)) = w((-x)+(-y))$ and $- \succ +$. The corresponding algebra $\mathcal{A}$ is given by:

$$x +_{\mathcal{A}} y = x + y + 2 \qquad -_{\mathcal{A}}(x) = x \qquad \mathsf{0}_{\mathcal{A}} = 0$$

For instance, $-(x + y) >_{\mathsf{wpo}} (-x) + (-y)$ is derived as follows:

$$\dfrac{-(x+y) \geq_{\mathcal{A}} (-x) + (-y) \qquad - \succ + \qquad \dfrac{-(x+y) >_{\mathcal{A}} -x}{-(x+y) >_{\mathsf{wpo}} -x} \qquad \cdots}{-(x+y) >_{\mathsf{wpo}} (-x) + (-y)}$$

As shown in Example 81, the class gwpo_{01} is strictly larger than the KBO. Also, $\mathsf{wpo}_{\mathbb{N}}$ is strictly larger than the KBO.

Example 88 (continued from Example 84)**.** The termination of the TRS is shown by $\mathsf{wpo}_{\mathbb{N}}$ but not by the KBO due to the duplicating rule $\mathsf{f}(\mathsf{s}(x), y) \to \mathsf{f}(x, \mathsf{s}(x))$.

The lexicographic path order. Even the subclass $\mathsf{gwpo}_{\mathbb{N}}$ does not subsume the LPO. This is because the simulation of the LPO via the WPO requires max-interpretation [34]: $f_{\mathcal{A}}(x_1, \ldots, x_n) = \max(x_1, \ldots, x_n)$ for each n-ary function symbol f. On the other hand, the subclasses $\mathsf{gwpo}_{\mathbb{N}}$ uses polynomials. The fact that polynomials cannot simulate max-interpretation is witnessed by the following example:

Example 89. The termination of SK90_4.23 in TPDB 11.0:

$$\begin{array}{rl}
\mathsf{if}(\mathsf{true}, x, y) \to x & \mathsf{if}(\mathsf{false}, x, y) \to y \\
\mathsf{if}(x, y, y) \to y & \mathsf{if}(\mathsf{if}(x, y, z), u, v) \to \mathsf{if}(x, \mathsf{if}(y, u, v), \mathsf{if}(z, u, v)) \\
\mathsf{if}(x, \mathsf{if}(x, y, z), z) \to \mathsf{if}(x, y, z) & \mathsf{if}(x, y, \mathsf{if}(x, y, z)) \to \mathsf{if}(x, y, z)
\end{array}$$

Consider linear polynomial interpretation $\mathcal{A}$ over $\mathbb{N}$, and let $\mathsf{if}_{\mathcal{A}}(x, y, z) = ax+by+cz+d$. To show the termination by $\mathsf{gwpo}_{\mathbb{N}}$, we need $\mathsf{if}(\mathsf{true}, x, y) \geq_{\mathcal{A}} x$

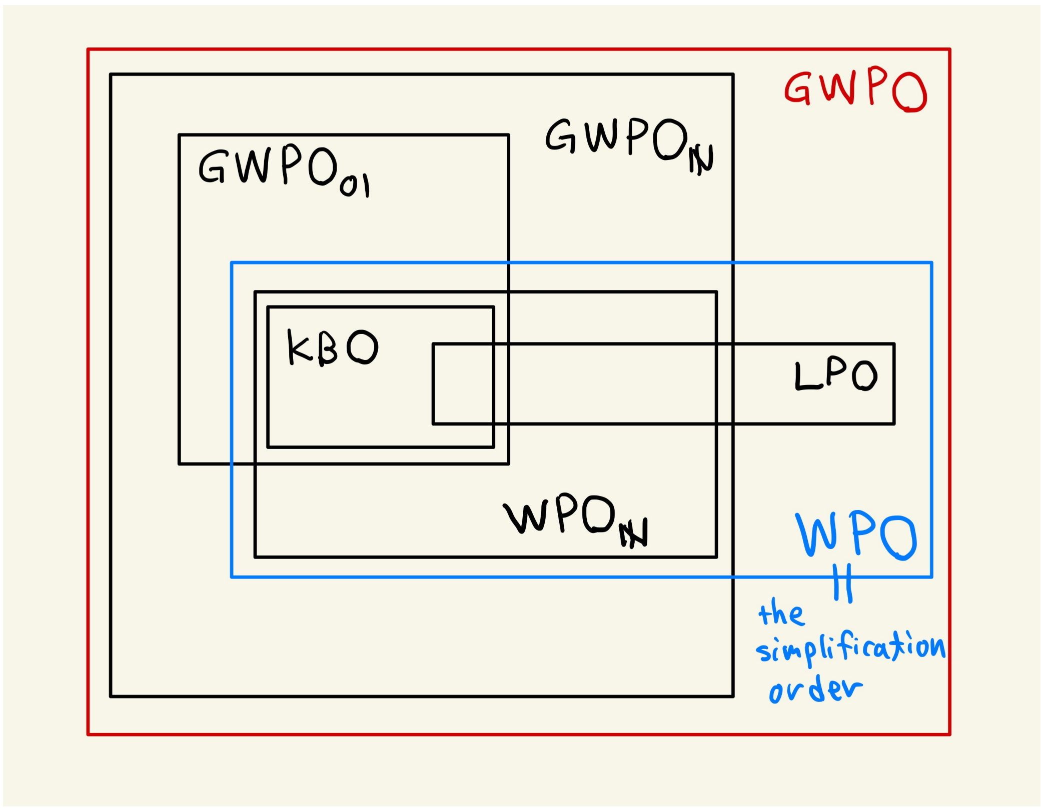

Figure 3.1: A hierarchy chart of classes of reduction orders.

and $\mathsf{if}(\mathsf{false}, x, y) \geq_{\mathcal{A}} y$, which yields $b, c \geq 1$. On the other hand, we also need $\mathsf{if}(\mathsf{if}(x, y, z), u, v) \geq_{\mathcal{A}} \mathsf{if}(x, \mathsf{if}(y, u, v), \mathsf{if}(z, u, v))$. By comparing the coefficients of u we obtain the constraint $b \geq b^2 + bc$, which is impossible under $b, c \geq 1$. Thus, we cannot show the termination by $\mathsf{gwpo}_{\mathbb{N}}$. Observe that, due to the presence of the embedding rules $\mathsf{if}(\mathsf{true}, x, y) \to x$ and $\mathsf{if}(\mathsf{false}, x, y) \to y$, polynomials are forced to be strictly monotone. However, the termination is easily shown by the LPO with the empty precedence.

3.4 Implementation

Before we proceed to experimental results, we introduce three subclasses of the GWPO and their implementation details implemented in Toma. Also, the implementation of the LPO in Toma is briefly discussed. In examples of each option, its input `problem.trs` is given the CoCo TRS format [24].

Encoding $\mathsf{gwpo}_{\mathbb{N}}$. Orientation via $\mathsf{gwpo}_{\mathbb{N}}$ is encoded in a naive way, following the recursive definition of $>_{\mathsf{wpo}'}$. In particular, the relations $>_{\mathcal{A}}$ and $\geq_{\mathcal{A}}$ via an algebra $\mathcal{A}$ are encoded using the following quantifier elimination for linear polynomials:

Proposition 90. *Let $n \in \mathbb{N}$ and $a_0, \dots, a_n, b_0, \dots, b_n \in \mathbb{N}$. The following statements are equivalent:*

- $a_0 + a_1 x_1 + \cdots + a_n x_n > b_0 + b_1 x_1 + \cdots + b_n x_n$ *for all* $x_1, \dots, x_n \in \mathbb{N}$.
- $a_0 > b_0$ *and* $a_k \geq b_k$ *for all* $k \in \{1, \dots, n\}$.

Similarly, the following statements are equivalent:

- $a_0 + a_1 x_1 + \cdots + a_n x_n \geq b_0 + b_1 x_1 + \cdots + b_n x_n$ *for all* $x_1, \dots, x_n \in \mathbb{N}$.
- $a_k \geq b_k$ *for all* $k \in \{0, \dots, n\}$.

Example 91. We explain how to encode the constraint for $\mathsf{f}(\mathsf{f}(x)) >_{\mathcal{A}} \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$. Let $\mathsf{f}_{\mathcal{A}}(x) = f_0 + f_1 x$ and $\mathsf{g}_{\mathcal{A}}(x) = g_0 + g_1 x$. Here f_0, f_1, g_0 and g_1 are unknown constants over $\mathbb{N}$. We have $\mathsf{f}_{\mathcal{A}}(\mathsf{f}_{\mathcal{A}}(x)) = (f_0 + f_1 f_0) + f_1 f_1 x$ and $\mathsf{f}_{\mathcal{A}}(\mathsf{g}_{\mathcal{A}}(\mathsf{f}_{\mathcal{A}}(x))) = (f_0 + f_1 g_0 + f_1 g_1 f_0) + f_1 g_1 f_1 x$ by calculation. By using the quantifier elimination, the orientation $\mathsf{f}(\mathsf{f}(x)) >_{\mathcal{A}} \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$ is encoded into the following constraints:

for the constant part: $f_0 + f_1 f_0 > f_0 + f_1 g_0 + f_1 g_1 f_0$

for x: $f_1 f_1 \geq f_1 g_1 f_1$

In the SMT-LIB language [6], the constraints are encoded into S-expressions as follows:

for the constant part:

```
(> (+ f0 (* f1 f0)) (+ f0 (* f1 g0) (* f1 g1 f0)))
```

for x: `(> (* f1 f1) (* f1 g1 f1))`

where `f0`,`f1`,`g0`, and `g1` have the type `Int`.

The termination checker using $\mathsf{gwpo}_{\mathbb{N}}$ is available via:

```
toma --termination problem.trs --gwpoN
```

Encoding gwpo_{01}. A problem of $\mathsf{gwpo}_{\mathbb{N}}$ is that the constraints for orientation $>_{\mathcal{A}}$ and $\geq_{\mathcal{A}}$ are based on non-linear arithmetic, such as:

```
(> (* f1 f1) (* f1 g1 f1))
```

As a result, SMT solvers do not solve constraints efficiently. In particular, the efficiency for finding a reduction order matters in the setting of theorem proving. This is why we introduce gwpo_{01}. For instance, Z3 offers if-then-else expressions in the form of `(ite e1 e2 e3)`, which evaluates `e2` if `e1` evaluates to `true`, or evaluates `e3` otherwise. We utilize this feature to encode $>_{\mathcal{A}}$ and $\geq_{\mathcal{A}}$ in linear arithmetic. An example of encoding $>_{\mathcal{A}}$ is given below:

Example 92 (continued from Example 91)**.** For instance, the multiplication `(* f1 g1 f0)` is encoded to:

```
(ite (and f1 g1) f0 0)
```

where `f0, g0` have the type `Int` but `f1` and `g1` have the type `Bool`. Here, we associate 1 and 0 with `true` and `false`, respectively. By this idea, the orientation $\mathsf{f}(\mathsf{f}(x)) >_{\mathcal{A}} \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))$ are encoded into the following SMT-LIB expressions:

for the constant part:

```
(> (+ f0 (ite f1 f0 0))
   (+ f0 (ite f1 g0 0) (ite (and f1 g1) f0 0)))
```

for x:

```
(> (ite (and f1 f1) 1 0)
   (ite (and f1 g1 f1) 1 0))
```

Now Z3 can solve the constraints efficiently, using its linear arithmetic solver.

The termination checking by gwpo_{01} is available via:

```
toma --termination problem.trs --gwpo01
```

Encoding $\mathsf{wpo}_{\mathbb{N}}$. The encoding of $\mathsf{wpo}_{\mathbb{N}}$ follows the way of $\mathsf{gwpo}_{\mathbb{N}}$. The termination checker via $\mathsf{wpo}_{\mathbb{N}}$ is available via:

```
toma --termination problem.trs --wpo
```

The LPO. The LPO is also implemented in Toma. If we implement the LPO in a naive way following its recursive definition, the time complexity is exponential with respect to the size of the two terms compared. Thus, we use bottom-up construction of constraints described in [21] to achieve polynomial-time comparison. The termination checker using the LPO is available via:

```
toma --termination problem.trs --lpo
```

3.5 Experiments

Now we are ready to discuss experimental results. We run the tools on the TPDB problems 11.0 [1] setting 60 seconds time limit on the machine equipped with the processor Intel Core i5-8365U CPU @ 1.60GHz and 8 GB memory. First we explain how to read the tables Table 3.1. The meaning of each status on an input TRS file is explained as follows:

YES	the tool successfully showed termination.
MAYBE	the tool gave up proof of termination before the 60 seconds time limit, including the case when there is no proof via the ordering.
TIMEOUT	the tool exceeded the 60 seconds time limit.
ERROR	the tool quitted due to an exception, such as an out-of-memory error.

For comparison, we use the KBO option of TTT2 [20] version 1.20 via

```
ttt2 -C "" -t -s 'kbo' problem.trs
```

and referred to as kbo in the experiment table Table 3.1. The complete results are available at:

https://www.jaist.ac.jp/~s2110079/termination/termination.html

Finally, we are ready to discuss the experimental result.

Simulation. The theoretical subsumption relations shown in Section 3.3 are almost observed in the experimental result. However, proper subsumption may not hold due to the 60 seconds time limit. For instance, the problem Transformed_CSR_04_PALINDROME_nokinds_Z is solved by the KBO in a second, but neither $\mathsf{gwpo}_{\mathbb{N}}$ nor $\mathsf{wpo}_{\mathbb{N}}$ is not able to solve the problem in 60 seconds. In contrast, gwpo_{01} subsumes the KBO even in the experiment, and the average time is slightly better than that of the KBO.

Table 3.1: Termination analysis on TPDB 11.0 (time in seconds).

status	$\mathsf{gwpo}_{\mathbb{N}}$	gwpo_{01}	$\mathsf{wpo}_{\mathbb{N}}$	lpo	kbo
YES	**316**	**286**	**174**	**144**	**79**
average time	*1.09*	*0.15*	*0.65*	*0.02*	*0.22*
MAYBE	1077	1196	1258	1353	1419
average time	*3.15*	*0.54*	*1.92*	*0.06*	*0.22*
TIMEOUT	101	12	63	1	0
average time	*60.00*	*60.00*	*60.00*	*60.00*	
ERROR	4	4	3	0	0
average time	*19.77*	*31.28*	*18.92*		

Comparison between $\mathsf{gwpo}_{\mathbb{N}}$ and gwpo_{01}. The subclass gwpo_{01} works more efficiently than $\mathsf{gwpo}_{\mathbb{N}}$, with a small cost of losing its power as a termination checker. For instance, the problem TCT_12_sat is solved by gwpo_{01}, so theoretically instances of $\mathsf{gwpo}_{\mathbb{N}}$ with the same parameters prove the termination of the TRS. But the implementation of $\mathsf{gwpo}_{\mathbb{N}}$ is not able to find such a parameter in 60 seconds. This is thanks to its encoding to linear arithmetic, where Z3 has an efficient solver. In particular, gwpo_{01} gives up MAYBE problems earlier, exhausting whole search space of the constraints. The results also show that competence of the ordering of gwpo_{01} in variants of maximal completion, where constraint solving of the ordering is called for each iteration of the procedure. However, we note that a typical constraint is described as a MaxSMT problem in maximal completion, in contrast to a constraint for termination checking, described as a pure SMT solving problem. The comparison in theorem proving is included in Chapter 4.

A memory issue of the GWPO. The implementations of $\mathsf{gwpo}_{\mathbb{N}}$, gwpo_{01} and $\mathsf{wpo}_{\mathbb{N}}$ exit with the status ERROR running out of available memory. For instance, such an issue is observed in the problem MNZ_10_5, which contains deeply nested terms, such as $\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(\mathsf{s}(0)))))))))) \to \mathsf{k}(\mathsf{s}(\mathsf{s}(0)))$. Because of their native implementations which simply follow the recursive definition, size of a constraint for nested terms exponentially explodes. This does not happen for the LPO, since it is carefully implemented to avoid the explosion. This issue implies necessity of an efficient encoding method for the GWPO.

Chapter 4

Maximal Ordered Completion with Simplification

In this chapter we extend maximal ordered completion by Winkler and Moser [33]. The extension allows *simplification* of equations during a run, which is only allowed after critical pair generation in the original maximal ordered completion. On the other hand, it is also an extension of standard ordered completion by Martin and Nipkow [23] in the sense that every run of standard ordered completion can be simulated by the new procedure. In other words, the extension is also considered as a variant of standard ordered completion that allows change of reduction orders during a run. Moreover, a new equational theorem proving procedure is obtained as a variant of the extended maximal ordered completion. These new procedures are implemented in an equational theorem prover Toma.

4.1 Abstract Ordered Completion

The following formalization of abstract ordered completion is due to [23].

Definition 93 ([23])**.** Let $>$ be a reduction order. We define the binary relation $\vdash$ on equational systems via the following inference rules:

deduce: $\mathcal{E} \vdash \mathcal{E} \cup \{s \approx t\}$ if $s \leftrightarrow_{\mathcal{E}} \cdot \leftrightarrow_{\mathcal{E}} t$

delete: $\mathcal{E} \uplus \{s \approx t\} \vdash \mathcal{E}$ if s and t are ground-joinable in $(\mathcal{E}, >)$

simplify:

- $\mathcal{E} \uplus \{s \approx t\} \vdash \mathcal{E} \cup \{u \approx t\}$ if $s \to_{(\mathcal{E},>)} u$
- $\mathcal{E} \uplus \{s \approx t\} \vdash \mathcal{E} \cup \{s \approx u\}$ if $t \to_{(\mathcal{E},>)} u$

We call the fragment of the inference rules without deduce *simplification*, and write $\vdash_{\mathsf{s}}$ for the induced binary relation.

Example 94. Consider the equational system $\mathcal{E}_{\div}$

$$\begin{aligned} \mathsf{0} - y &\approx \mathsf{0} & \mathsf{0} \div \mathsf{s}(y) &\approx \mathsf{0} \\ x - \mathsf{0} &\approx x & \mathsf{s}(x) \div \mathsf{s}(y) &\approx \mathsf{s}((x-y) \div \mathsf{s}(y)) \\ \mathsf{s}(x) - \mathsf{s}(y) &\approx x - y \end{aligned}$$

plus another equation $\mathsf{s}(\mathsf{0}) \div \mathsf{s}(y) \approx \mathsf{s}(\mathsf{0})$. The additional equation is redundant, as we can simulate reasoning by other rules. Simplification removes such an equation; consider the GWPO $>_{\mathsf{gwpo}}$ of Example 81, and it orients every equation in $\mathcal{E}_{\div}$ from left to right. So the following rewrite sequence via the ordered rewrite system $(\mathcal{E}_{\div}, >_{\mathsf{gwpo}})$ is valid:

$$\mathsf{s}(\mathsf{0}) \div \mathsf{s}(y) \to \mathsf{s}((\mathsf{0} - y) \div \mathsf{s}(y)) \to \mathsf{s}(\mathsf{0} \div \mathsf{s}(y)) \to \mathsf{s}(\mathsf{0})$$

Applying the simplify rule three times, and then the delete rule that removes $\mathsf{s}(\mathsf{0}) \approx \mathsf{s}(\mathsf{0})$, we obtain $\mathcal{E}_{\div} \uplus \{\mathsf{s}(\mathsf{0}) \div \mathsf{s}(y) \approx \mathsf{s}(\mathsf{0})\} \vdash_{\mathsf{s}}^4 \mathcal{E}_{\div}$.

We prove that abstract ordered completion preserves conversion equivalence on ground terms. The following is a convenient lemma.

Lemma 95. *Let $\mathcal{E}_1$ and $\mathcal{E}_2$ be equational systems. Suppose that every ground instance of each equation in $\mathcal{E}_1$ is convertible in $\mathcal{E}_2$, and every ground instance of each equation in $\mathcal{E}_2$ is convertible in $\mathcal{E}_1$. Then $\mathcal{E}_1$ and $\mathcal{E}_2$ are conversion equivalent on ground terms.*

Proof. If the signature contains no constant, there is no ground term. In this case, the conversions on ground terms collapse to the empty relation, and thus the equivalence is vacuously satisfied. So for the rest of proof, we assume that the signature contains a constant symbol. Let s and t be ground terms, and suppose $s \leftrightarrow_{\mathcal{E}_1}^* t$. Substituting each variable in the conversion with a constant, we obtain another conversion between s and t such that only ground terms appear in the conversion. At this point, each one-step conversion in the new conversion is a ground instance of an equation in $\mathcal{E}_2$. Applying the assumption, we obtain $s \leftrightarrow_{\mathcal{E}_2}^* t$. The same argument applies to the other direction. □

Proposition 96. *If $\mathcal{E}_1 \vdash^* \mathcal{E}_2$, then $\leftrightarrow_{\mathcal{E}_1}^*$ and $\leftrightarrow_{\mathcal{E}_2}^*$ coincide on ground terms.*

Proof. It suffices to show the claim for an arbitrary one-step derivation $\mathcal{E}_1 \vdash \mathcal{E}_2$. Analyzing the applied inference rule, we obtain the following inclusions:

deduce: $\mathcal{E}_1 \subseteq \mathcal{E}_2$ and $\mathcal{E}_2 \subseteq \mathcal{E}_1 \cup (\leftrightarrow_{\mathcal{E}_1} \cdot \leftrightarrow_{\mathcal{E}_1})$

delete: $\mathcal{E}_1 \subseteq \mathcal{E}_2 \cup \Downarrow_{(\mathcal{E}_2,>)}$ and $\mathcal{E}_2 \subseteq \mathcal{E}_1$

simplify:

- $\mathcal{E}_1 \subseteq \leftrightarrow_{\mathcal{E}_2}^{=} \cdot \mathcal{E}_2$ and $\mathcal{E}_2 \subseteq \leftrightarrow_{\mathcal{E}_1}^{=} \cdot \mathcal{E}_1$
- $\mathcal{E}_1 \subseteq \mathcal{E}_2 \cdot \leftrightarrow_{\mathcal{E}_2}^{=}$ and $\mathcal{E}_2 \subseteq \mathcal{E}_1 \cdot \leftrightarrow_{\mathcal{E}_1}^{=}$

Thus the claim follows immediately from Lemma 95. □

The relation $\vdash_{\mathsf{s}}$ is well-founded for all reduction orders.

Proposition 97. *The relation $\vdash_{\mathsf{s}}$ is well-founded for all reduction orders.*

Proof. The rule delete decreases the number of equations $|\mathcal{E}|$ and the rule simplify decreases the multiset $\bigcup\{\{s,t\} \mid s \approx t \in \mathcal{E}\}$ with respect to the multiset extension of the reduction order. Combine these two measures lexicographically. □

4.2 Extending Maximal Ordered Completion

Definition 98. The procedure $\mathsf{P1}$, *maximal ordered completion with simplification*, takes a set of equations $\mathcal{E}$ as an input, and returns an OTRS as an output, and its algorithm is given by Procedure 1. Here $\mathcal{C}$ denotes a set of equations and $>$ denotes a reduction order. The procedures $O(\mathcal{C})$, $\psi(\mathcal{C},>)$ and $S(\mathcal{C},>)$ are described below:

- The procedure $O(\mathcal{C})$ is an arbitrary procedure that returns a reduction order.
- The procedure $\psi(\mathcal{C},>)$ performs simplification $\vdash_{\mathsf{s}}^*$ on the set $\mathcal{C}$ of equations using the reduction order $>$, and returns the simplified version of $\mathcal{C}$.
- The procedure $S(\mathcal{C},>)$ is an arbitrary procedure that returns a set of equations in $\leftrightarrow_{\mathcal{E}}^*$.

To obtain a concrete procedure from Definition 98, we need to instantiate O, ψ S, ground joinability testing $\Downarrow$, and ground confluence testing. The procedure $\mathsf{P1}$ does not fall into deadlock due to simplification because the termination of simplification is guaranteed by Proposition 97. For the correctness proof, we need a lemma for ground-equivalence.

Procedure 1 Maximal Ordered Completion with Simplification

```
1: procedure P1($\mathcal{E}$)
2:     $\mathcal{C} := \mathcal{E}$
3: loop:
4:     $> := O(\mathcal{C})$
5:     $\mathcal{C}' := \psi(\mathcal{C}, >)$
6:     if $\mathcal{E} \subseteq \Downarrow_{(\mathcal{C}',>)}$ and $(\mathcal{C}', >)$ is ground-confluent then return $(\mathcal{C}', >)$
7:     $\mathcal{C} := \mathcal{C}' \cup S(\mathcal{C}', >)$
8:     goto loop
```

Lemma 99. *Let $\mathcal{E}$ and $\mathcal{E}'$ be ground-equivalent equational systems, and $>$ be a reduction order. If $\mathcal{E} \subseteq \Downarrow_{(\mathcal{E}',>)}$, then the equational system $\mathcal{E}$ and the OTRS $(\mathcal{E}', >)$ are ground-equivalent.*

Proof. By the same argument as Lemma 95, without loss of generality we can assume that there is a constant in the signature. The inclusion $\leftrightarrow^*_{(\mathcal{E}',>)} \subseteq \leftrightarrow^*_{\mathcal{E}}$ on ground terms follows from the trivial inclusion $\leftrightarrow^*_{(\mathcal{E}',>)} \subseteq \leftrightarrow^*_{\mathcal{E}'}$ and the ground equivalence of $\mathcal{E}$ and $\mathcal{E}'$. For the other direction, suppose $s \leftrightarrow^*_{\mathcal{E}} t$ for ground terms s, t. Substituting every variable in the conversion we have another conversion between s and t such that every term in the conversion is ground. Now we apply the assumption $\mathcal{E} \subseteq \Downarrow_{(\mathcal{E}',>)}$, and obtain a conversion $s \leftrightarrow^*_{(\mathcal{E}',>)} t$. □

The following example shows why $\mathcal{E} \subseteq \Downarrow_{(\mathcal{E}',>)}$ in Lemma 99 cannot be dropped.

Example 100. Let $\mathcal{E} = \{\mathsf{a} \approx \mathsf{b}\}$, and $>_{\mathsf{lpo}}$ be the LPO induced by the empty precedence. The OTRS $\mathcal{O} = (\mathcal{E}, >_{\mathsf{lpo}})$ is not ground-equivalent to $\mathcal{E}$, as $\mathsf{a} \to_{\mathcal{E}} \mathsf{b}$ but $\mathsf{a}, \mathsf{b} \in \mathsf{NF}(\mathcal{O})$.

Proposition 101. *Let $\mathcal{E}$ be an equational system. Suppose that the run $\mathsf{P1}(\mathcal{E})$ terminates and returns an OTRS $(\mathcal{E}', >)$. Then $(\mathcal{E}', >)$ is a ground-complete presentation for $\mathcal{E}$.*

Proof. Let $\mathcal{E}$ be an input ES and $(\mathcal{E}', >)$ the output. The ground termination is guaranteed by the reduction order $>$, and the ground confluence follows from the termination condition. For the ground equivalence of $\mathcal{E}$ and the OTRS $(\mathcal{E}', >)$ first we show ground equivalence of the equational systems $\mathcal{E}$ and $\mathcal{E}'$. We appeal to Lemma 95. We have $\mathcal{E} \subseteq \Downarrow_{(\mathcal{E}',>)}$ from the termination condition. The other direction is easily shown by following the algorithm using Proposition 96 and $S(\mathcal{C}, >) \subseteq \leftrightarrow^*_{\mathcal{E}}$. Now ground equivalence of $\mathcal{E}$ and the OTRS $(\mathcal{E}', >)$ follows from lemma 99 and the termination condition. □

The procedure P1 can simulate ordered completion Definition 93 by taking O that returns a fixed reduction order.

Proposition 102 (simulation)**.** *Let $\mathcal{E} \vdash^* \mathcal{E}'$ be a run of abstract ordered completion with a reduction order $>$. If $(\mathcal{E}', >)$ is ground-confluent, then there is an instance of the procedure* P1 *such that* $\mathsf{P1}(\mathcal{E}) = \mathcal{E}'$.

Proof. We can achieve $\mathsf{P1}(\mathcal{E}) = \mathcal{E}'$ by programming O, ψ and S as follows: O always returns $>$. For ψ and S, factorize $\mathcal{E} \vdash^* \mathcal{E}'$ into deduce and $\vdash_{\mathsf{s}}$, and program ψ and S to perform each consecutive steps at each iteration. The other termination condition $\mathcal{E} \subseteq \Downarrow_{(\mathcal{E}',>)}$ follows from the ground equivalence and ground confluence. $\square$

Next we give an equational theorem proving procedure based on Procedure 2.

Definition 103 (theorem proving via maximal ordered completion)**.** The procedure P2, a theorem proving procedure via maximal ordered completion, takes a set of equations $\mathcal{E}$ and a ground equation $s \approx t$ as an input, and returns *true* or *false* as an output, and its procedure is given by Procedure 2. The procedures $O(\mathcal{C}, s \approx t)$, $\psi(\mathcal{C}, >)$ and $S(\mathcal{C}, >)$ are given in the same way as definition 98.

Procedure 2 Theorem Proving Variant of P1

```
1: procedure P2($\mathcal{E}$, $s \approx t$)
2:     $\mathcal{C} := \mathcal{E}$
3: loop:
4:     $> := O(\mathcal{C}, s \approx t)$
5:     $\mathcal{C}' := \psi(\mathcal{C}, >)$
6:     if $s \downarrow_{(\mathcal{C}',>)} t$ then return true
7:     if $\mathcal{E} \subseteq \Downarrow_{(\mathcal{C}',>)}$ and $(\mathcal{C}', >)$ is ground-confluent then return false
8:     $\mathcal{C} := \mathcal{C}' \cup S(\mathcal{C}', >)$
9:     goto loop
```

In contrast to P1, the procedure O additionally takes the goal $s \approx t$ as an input to make the procedure more goal-oriented. The next proposition states that the theorem proving procedure P2 is correct.

Proposition 104. *If* P2 *returns true, the ground equation $s \approx t$ is valid in $\mathcal{E}$. If* P2 *returns false, the ground equation $s \approx t$ is invalid in $\mathcal{E}$.*

Proof. The statement is shown in the same manner as Proposition 101 using Proposition 53. $\square$

4.3 Implementation of Toma

Note that the definitions Definition 98 and Definition 103 do not specify O and S in the definition. In this section, we describe how to instantiate O and S, and give further details of the implementation of Toma, an equational theorem prover based on maximal ordered completion with simplification. At the moment Toma only supports the LPO and the GWPO as its reduction order.

Finding a Reduction Order. The procedure O returns a reduction order given an ES $\mathcal{C}$, and a goal $s \approx t$ in the case of P2. To implement O, we follow the approach of MædMax [33], namely maximization of the reducibility of the ES $\mathcal{C}$ as an ordered rewrite system. The reducibility of an ordered rewrite system $(\mathcal{C}, >)$ is, for instance, measured by the set of reducible terms $\{t \in \mathcal{T}(\mathcal{F}, \mathcal{V}) \mid t \notin \mathsf{NF}(\mathcal{C}, >)\}$. However, for practical reasons, we use the following finite approximation proposed by Winkler and Moser [33]: firstly we limit $\mathcal{T}(\mathcal{F}, \mathcal{V})$ to a finite set of terms. In case of P1, we use $T = \bigcup\{\{\ell, r\} \mid \ell \approx r \in \mathcal{C}\}$ instead of $\mathcal{T}(\mathcal{F}, \mathcal{V})$, as a test set of reducibility. In case of P2, we add the terms s and t in the goal to the test set T, to archive more goal-oriented search of a reduction order. Secondly the reducibility condition $t \notin \mathsf{NF}(\mathcal{C}, >)$ is approximately encoded to the formula $R(t)$:

$$R(t) \equiv \exists \ell \exists r (\ell \approx r \in \mathcal{C} \cup \mathcal{C}^{-1} \land t \trianglerighteq \ell \land \ell > r)$$

In contrast to the directly encoded formula of $t \notin \mathsf{NF}(\mathcal{C}, >)$

$$\exists \ell \exists r \exists u \exists \sigma (\ell \approx r \in \mathcal{C} \cup \mathcal{C}^{-1} \land u \trianglelefteq t \land \ell\sigma = u \land \ell\sigma > r\sigma)$$

the size of the constraints is feasibly small, thanks to the elimination of the quantifications $\exists u$ and $\exists \sigma$. Using the approximations above, we solve the maximizaton problem of the number $|\{t \in T \mid R(t)\}|$ via Z3 [12], which provides maxSMT solving. Encoding of the LPO and the GWPO follows the description in Chapter 3. In particular, Toma imposes ground-totality to precedences of the LPO so that induced LPOs are ground-total. This requirement is essential when we use the Extended Critical Pair Lemma in ground-confluence testing, as discussed later.

Critical Pair Selection. The procedure $S(\mathcal{C}, >)$ returns a set of critical pairs:

$$\{s \approx t \in \mathsf{ECP}(\mathcal{C}, >) \mid s \approx t \notin \mathcal{C} \cup \mathcal{C}^{-1},\ s \text{ and } t \text{ are not joinable in } (\mathcal{C}, >)\}$$

Currently we do not restrict the maximum number of equations in $S(\mathcal{C}, >)$.

Ground Joinability Testing. Given an equation $s \approx t$ and an ordered rewrite system $(\mathcal{C}, >)$, check the ground joinability $s \Downarrow_{(\mathcal{C},>)} t$ in two steps:

1. If the equation is joinable $s \downarrow_{(\mathcal{C},>)} t$, obviously it is ground-joinable.
2. Otherwise, additionally the tool uses Martin and Nipkow's ground joinability testing when $>$ is an LPO.

Currently, Toma has no additional ground joinability testing for the GWPO.

Ground Confluence Testing. For ground confluence of an OTRS equipped with an LPO, Toma uses the Extended Critical Pair Lemma. The requirement for ground-totality is satisfied as Toma only considers total precedences. In contrast, we cannot use the Extended Critical Pair Lemma for the GWPO since an instance of the GWPO is not necessarily ground-total. For this case, we utilize the Critical Pair Lemma as confluence trivially implies ground confluence. We say an OTRS $(\mathcal{E}, >)$ is a TRS if every $s \approx t \in \mathcal{E}$ satisfies $s > t$ or $t > s$. Toma checks ground confluence an OTRS $(\mathcal{E}, >)$ in two steps. It first verify that $(\mathcal{E}, >)$ is a TRS, and every extended critical pair of $(\mathcal{E}, >)$ is joinable in $(\mathcal{E}, >)$. The correctness of this test is shown by combination of the Critical Pair Lemma and the following proposition.

Proposition 105. *Let $(\mathcal{E}, >)$ be an OTRS. Suppose that the OTRS $(\mathcal{E}, >)$ is a TRS. Then the TRS $\mathcal{E}^{>} = \{s \to t \mid s \approx t \in \mathcal{E} \cup \mathcal{E}^{-}, s > t\}$ satisfies $\mathsf{ECP}(\mathcal{E}, >) = \mathsf{CP}(\mathcal{E}^{>})$, and $\to_{(\mathcal{E},>)}$ and $\to_{\mathcal{E}^{>}}$ coincide.*

The Goal Transformation. The procedure P2 only takes a ground equation as a goal. We transform an arbitrary goal $s \approx t$ to a ground goal a la Waldmeister [22]: add equations $\mathsf{eq}(x, x) = \mathsf{T}$ and $\mathsf{eq}(s, t) = \mathsf{F}$ to the initial ES, and set $\mathsf{T} = \mathsf{F}$ as a goal. Here eq, T and F are fresh function symbols. This transformation is known to be complete: $s \approx t$ is valid in the initial ES if and only if $\mathsf{T} \approx \mathsf{F}$ is valid in the transformed ES.

Let us illustrate how Toma works by an example.

Example 106. We follow how Toma complete the equational system $\mathcal{E}_{\div}$

$$\begin{aligned} \mathsf{0} - y &\approx \mathsf{0} & \mathsf{0} \div \mathsf{s}(y) &\approx \mathsf{0} \\ x - \mathsf{0} &\approx x & \mathsf{s}(x) \div \mathsf{s}(y) &\approx \mathsf{s}((x - y) \div \mathsf{s}(y)) \\ \mathsf{s}(x) - \mathsf{s}(y) &\approx x - y \end{aligned}$$

using P1 with gwpo_{01}. First it finds an instance of gwpo_{01} that maximizes its reducibility. The reducibility is measured upon the set T_1 of all terms in $\mathcal{E}_{\div}$. For this case we have nine terms:

$$T_1 = \{\mathsf{0} - y, \mathsf{0}, x - \mathsf{0}, x, \mathsf{s}(x) - \mathsf{s}(y), x - y, \mathsf{0} \div \mathsf{s}(y), \mathsf{s}(x) \div \mathsf{s}(y), \mathsf{s}((x - y) \div \mathsf{s}(y))\}$$

We orient each equation in $\mathcal{E}_{\div}$ so that maximize the number of reducible terms in T_1. One solution is the parameter in Example 81, which orients every equation from left to right. In this solution, five terms in T_1 is reducible, and this choice yields a complete presentation for $\mathcal{E}_{\div}$. Another solution is to orient $\mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y))$ from right to left, and other equations from left to right.

$$\begin{aligned} \mathsf{0} - y &> \mathsf{0} & \mathsf{0} \div \mathsf{s}(y) &> \mathsf{0} \\ x - \mathsf{0} &> x & \mathsf{s}(x) \div \mathsf{s}(y) &< \mathsf{s}((x - y) \div \mathsf{s}(y)) \\ \mathsf{s}(x) - \mathsf{s}(y) &> x - y \end{aligned}$$

This orientation is possible by choosing the precedence $\div \succ \mathsf{s} \succ - \succ \mathsf{0}$ and the algebra:

$$\mathsf{0}_{\mathcal{A}} = 1 \qquad \mathsf{s}_{\mathcal{A}}(x) = x + 1 \qquad x -_{\mathcal{A}} y = x + 2 \qquad x \div_{\mathcal{A}} y = 1$$

For instance, $\mathsf{s}((x-y) \div \mathsf{s}(y)) >_{\mathsf{gwpo}} \mathsf{s}(x) \div \mathsf{s}(y)$ holds; let $\ell = \mathsf{s}((x-y) \div \mathsf{s}(y))$ and $r = \mathsf{s}(x) \div \mathsf{s}(y)$. We have $\ell >_{\mathcal{A}} r$ as ℓ evaluates to 2 and r to 1, and $\ell >_{\mathsf{wpo}'} r$ is shown as follows:

$$\dfrac{\ell >_{\mathcal{A}} r \qquad \dfrac{\dfrac{\dfrac{x - y >_{\mathcal{A}} \mathsf{s}(x) \qquad \dfrac{x = x}{x - y >_{\mathsf{wpo}'} x}}{x - y >_{\mathsf{wpo}'} \mathsf{s}(x)}}{\vdots}}{\ell >_{\mathsf{wpo}'} \mathsf{s}(x)} \qquad \dfrac{\vdots}{\ell >_{\mathsf{wpo}'} \mathsf{s}(y)}}{\ell >_{\mathsf{wpo}'} r}$$

So we have $\ell >_{\mathsf{gwpo}} r$. Let us proceed with this GWPO $>_{\mathsf{gwpo}}$. The simplification step with ψ does nothing, as every equation is already simplified. In this case, the OTRS $(\mathcal{E}_{\div}, >_{\mathsf{gwpo}})$ does not satisfy the termination condition because it has non-joinable extended critical pairs, as we see in the next step. So Toma proceeds to critical pair generation. The admits seven extended critical pairs.

$$\begin{aligned} \mathsf{s}(x) \div \mathsf{s}((y - z) \div \mathsf{s}(z)) &\approx \mathsf{s}((x - ((y - z) \div \mathsf{s}(z))) \div (\mathsf{s}(y) \div \mathsf{s}(z))) \\ x - ((y - z) \div \mathsf{s}(z)) &\approx \mathsf{s}(x) - (\mathsf{s}(y) \div \mathsf{s}(z)) \\ ((x - y) \div \mathsf{s}(y)) - z &\approx (\mathsf{s}(x) \div \mathsf{s}(y)) - \mathsf{s}(z) \\ \mathsf{s}(\mathsf{s}(x)) \div \mathsf{s}(\mathsf{s}(y)) &\approx \mathsf{s}((x - y) \div \mathsf{s}(\mathsf{s}(y))) \\ \mathsf{0} \div (\mathsf{s}(x) \div \mathsf{s}(y)) &\approx \mathsf{0} \\ \mathsf{s}(x) \div \mathsf{s}(\mathsf{0}) &\approx \mathsf{s}(x \div \mathsf{s}(\mathsf{0})) \\ \mathsf{s}(\mathsf{0}) \div \mathsf{s}(y) &\approx \mathsf{s}(\mathsf{0} \div \mathsf{s}(y)) \end{aligned}$$

For instance, the second critical pair $x - ((y - z) \div \mathsf{s}(z)) \approx \mathsf{s}(x) - (\mathsf{s}(y) \div \mathsf{s}(z))$ is generated from the extended overlap:

$$(\mathsf{s}((x - y) \div \mathsf{s}(y)) \approx \mathsf{s}(x) \div \mathsf{s}(y), 2, \mathsf{s}(x) - \mathsf{s}(y) \approx x - y)$$

This extended overlap yields the following peak:

$$\mathsf{s}(x) - (\mathsf{s}(y) \div \mathsf{s}(z)) \leftarrow \mathsf{s}(x) - \mathsf{s}((y - z) \div \mathsf{s}(z)) \rightarrow x - ((y - z) \div \mathsf{s}(z))$$

Toma proceeds to the next iteration, and finds a reduction order for the twelve equations (five from the original ES, and seven from the extended critical pairs). One possible choice is the precedence $\mathsf{0} \succ \div \succ \mathsf{s} \succ -$ and the algebra $\mathcal{B}$ given by:

$$\mathsf{0}_{\mathcal{B}} = 30869 \qquad \mathsf{s}_{\mathcal{B}}(x) = x + 1 \qquad x -_{\mathcal{B}} y = x + 1 \qquad x \div_{\mathcal{B}} y = x + y + 1$$

In fact, these parameters are mechanically found by Toma. Next the tool runs simplification on the equational system using the GWPO. As a result, all the generated extended critical pairs are eliminated, and Toma obtains the original equational system $\mathcal{E}_{\div}$. For this case, the OTRS satisfies the termination condition because this GWPO orients $\mathcal{E}_{\div}$ as follows:

$$\begin{array}{rlrl}
\mathsf{0} - y >_{\mathsf{gwpo}} \mathsf{0} & & \mathsf{0} \div \mathsf{s}(y) >_{\mathsf{gwpo}} \mathsf{0} \\
x - \mathsf{0} >_{\mathsf{gwpo}} x & & \mathsf{s}(x) \div \mathsf{s}(y) >_{\mathsf{gwpo}} \mathsf{s}((x - y) \div \mathsf{s}(y)) \\
\mathsf{s}(x) - \mathsf{s}(y) >_{\mathsf{gwpo}} x - y & &
\end{array}$$

Completeness of this OTRS is shown in Example 46. Finally, Toma terminates and returns the OTRS.

4.4 Experiments in Ordered Completion

We evaluate the ordered completion procedure P1 using the mkbTT problems, which are used in the evaluation of maximal completion tools [18][33][13]. The result for this problem set is Table 4.1. In addition, we arrange another problem set containing non-simply terminating TRSs from the TPDB problems [1], such as ones with the label AG. The problems are from the collection of term rewrite systems by Aarts and Giesl [2]. In this experiment, tools run completion on TRSs by considering them as equational systems. Next we give configurations used in the experiments. In the following examples, `problem.trs` is given in the CoCo TRS format [24]. Tools are run on the machine equipped with the processor Intel Core i5-8365U CPU @ 1.60GHz and 8 GB memory.

- The configuration lpo is an implementation of P1 with $\psi(\mathcal{C}, >) = \mathcal{C}$ and O that returns an LPO, which is available via:

```
toma --ordered-completion problem.trs --lpo
        --no-inter-reduction
```

- The configuration lpo $+ \psi$ is an implementation of P1 with ψ that performs simplification, and O that returns an LPO, which is available via:

```
toma --ordered-completion problem.trs --lpo
```

- The configuration gwpo_{01} is an implementation of P1 with $\psi(\mathcal{C}, >) = \mathcal{C}$ and O that returns an instance of gwpo_{01}, which is available via:

```
toma --ordered-completion problem.trs --gwpo01
        --no-inter-reduction
```

- The configuration $\mathsf{gwpo}_{01} + \psi$ is an implementation of P1 with ψ that performs simplification, and O that returns an instance of gwpo_{01}, which is available via:

```
toma --ordered-completion problem.trs --gwpo01
```

- MædMax is run as a pure ordered completion tool, available via:

```
maedmax --complete-if-no-goal problem.trs
```

We are not aware of an option to execute Twee as a pure ordered completion tool. The meaning of each status is described as follows:

completed: a ground-complete presentation for the input is successfully obtained.

timeout: the run exceeded the 60 seconds time limit.

error: the run quitted due to an exception, such as an out-of-memory error or a parse error.

The complete result for Table 4.1 is available at

```
https://www.jaist.ac.jp/~s2110079/ordered-completion/
        ordered-completion.html
```

and one for Table 4.2 is available at:

```
https://www.jaist.ac.jp/~s2110079/AG01/AG01.html
```

Now we are ready to discuss on the results.

Table 4.1: The mkbTT equational systems (time in seconds).

status	lpo	$\text{lpo} + \psi$	gwpo_{01}	$\text{gwpo}_{01} + \psi$	MædMax
completed	46	61	31	52	88
average time	*2.60*	*2.23*	*2.09*	*2.33*	*0.93*
timeout	69	54	81	53	27
average time	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*
error	0	0	3	10	0
average time			*16.48*	*25.67*	

Table 4.2: A collection of TRSs from [2] (time in seconds).

status	lpo	$\text{lpo} + \psi$	gwpo_{01}	$\text{gwpo}_{01} + \psi$	MædMax
completed	15	14	18	19	20
average time	*0.45*	*0.05*	*2.27*	*4.22*	*0.11*
timeout	35	36	31	28	30
average time	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*
error	0	0	1	3	0
average time			*25.64*	*12.14*	

Comparison to MædMax. The numbers of completed equational systems by MædMax are the highest for the both problem sets. However, several problems are solved by Toma, which are not solved by MædMax. Such problems require use of non-simplification orders to find complete TRSs. For instance, the problem AG01_#3.1, another equational system for round-up division, is completed by gwpo_{01}, but not by MædMax. The problem is given as follows:

$$\begin{array}{cc} x - \mathsf{0} \approx x & \mathsf{0} \div \mathsf{s}(y) \approx \mathsf{0} \\ \mathsf{s}(x) - \mathsf{s}(y) \approx x - y & \mathsf{s}(x) \div \mathsf{s}(y) \approx \mathsf{s}((x - y) \div \mathsf{s}(y)) \end{array}$$

By orienting each equation from left to right by gwpo_{01} we obtain a complete presentation, which cannot be obtained by any simplification orders. This example show uniqueness of ordered completion by the GWPO. However, configurations with the GWPO quit on a couple of problems due to a memory error, exhausting available memory.

Improvement of performance by simplification. Generally, simplification improves average time required to complete systems, and the numbers of completed equational systems. However, it is to be noted that a configuration without ψ can solve a problem that is not solved by one with ψ. For instance, the problem AG01_#3.6a is completed by gwpo_{01}, but not by $\mathsf{gwpo}_{01} + \psi$. This is due to incompleteness of Toma's heuristics to find a reduction order. Given a word problem, Toma encode choice of a reduction order to a MaxSMT problem so that maximizes reducibility, but solutions of such an optimization problem may not be unique. This means that choice of a reduction order can depend on constraints generated by Toma which is highly influenced by presence of ψ, and on which reduction order is found by an employed MaxSMT solver. This shows limitation of the heuristics, maximization of reducibility.

4.5 Experiments in Theorem Proving

In this section, we evaluate the theorem proving procedure P2 for the word problem using the TPTP problems version 7.5.0 [31]. We build the set of word problems with the affirmative answer by the utility script `tptp2T`:

```
./tptp2T UnitEquality Status Unsatisfiable Form CNF
```

Similarly, we build the set of word problems with the negative answer by:

```
./tptp2T UnitEquality Status Satisfiable Form CNF
```

Next we give configurations used in the experiments. In the following examples, `problem.p` is given as a TPTP UEQ problem, which is interpreted as a word problem of an equational theory. Tools are run on the machine equipped with the processor Intel Core i5-8365U CPU @ 1.60GHz and 8 GB memory.

- The configuration lpo is an implementation of $\mathsf{P2}$ with $\psi(\mathcal{C}, >) = \mathcal{C}$ and O that returns an LPO, which is available via:

```
toma --waldmeister problem.p --lpo --no-inter-reduction
```

- The configuration $\mathsf{lpo} + \psi$ is an implementation of $\mathsf{P2}$ with ψ that performs simplification, and O that returns an LPO, which is available via:

```
toma --waldmeister problem.p --lpo
```

- The configuration gwpo_{01} is an implementation of $\mathsf{P2}$ with $\psi(\mathcal{C}, >) = \mathcal{C}$ and O that returns an instance of gwpo_{01}, which is available via:

```
toma --waldmeister problem.p --gwpo01 --no-inter-reduction
```

- The configuration $\mathsf{gwpo}_{01} + \psi$ is an implementation of $\mathsf{P2}$ with ψ that performs simplification, and O that returns an instance of gwpo_{01}, which is available via:

```
toma --waldmeister problem.p --gwpo01
```

- MædMax is run with the default option, available via:

```
maedmax problem.p
```

- Twee is run with the TSTP option, available via:

```
twee problem.p --tstp
```

The results of the options above are given in Table 4.3 and Table 4.4 The meaning of each status is described as follows:

solved: the problem is successfully solved by the configuration.

timeout: the configuration exceeded the 60 seconds time limit.

error: the configuration quitted due to an exception, such as an out-of-memory error or a parse error.

The complete result for Table 4.3 is available at

`https://www.jaist.ac.jp/~s2110079/ueq-unsat/ueq-unsat.html`

and one for Table 4.4 is available at:

`https://www.jaist.ac.jp/~s2110079/ueq-sat/ueq-sat.html`

Now we are ready to discuss on the results.

Comparison with MædMax. For problems with the affirmative answer, almost all problems solved by Toma are solved by MædMax, except for a few problems. Such a problem is, for instance, the problem ALG030-10, which is solved by the configuration $\mathsf{gwpo}_{01} + \psi$ but not by MædMax. For problems with the negative answer, the configuration $\mathsf{lpo} + \psi$ outperforms MædMax, thanks to simplification ψ. In particular, simplification works effectively for problems with many axioms, such as problems with the label NLP. For instance, the problem NLP002-10 with 55 axioms is not solved by MædMax, but the problem is solved by $\mathsf{lpo}+\psi$, producing a complete TRS with almost one hundred rules.

Table 4.3: Word problems with the affirmative answer (time in seconds).

status	lpo	$\mathsf{lpo} + \psi$	gwpo_{01}	$\mathsf{gwpo}_{01} + \psi$	MædMax	Twee
solved	158	177	151	171	625	794
average time	*7.74*	*4.70*	*7.64*	*7.04*	*5.02*	*1.64*
timeout	898	879	900	856	387	264
average time	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*
error	2	2	7	31	46	0
average time	*0.01*	*0.01*	*31.12*	*15.28*	*2.97*	

Comparison with Twee. Every problem solved by Toma is also solved by Twee, which implements the standard ordered completion with the KBO. The huge difference between the numbers of solved problems is attributed to the difference of its implementation; while Toma is currently naively implemented, Twee implements several optimization techniques such as term indexing [28] and its own redundancy criteria [30]. The implementation Toma need to be refined for fair comparison between ordered completion and maximal ordered completion with simplification.

Table 4.4: Word problems with the negative answer (time in seconds).

status	lpo	$\mathsf{lpo} + \psi$	gwpo_{01}	$\mathsf{gwpo}_{01} + \psi$	MædMax	Twee
solved	38	61	34	35	40	115
average time	*0.84*	*9.96*	*1.03*	*0.10*	*1.11*	*0.86*
timeout	219	196	223	218	183	144
average time	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*	*60.00*
error	2	2	2	6	36	0
average time	*0.01*	*0.01*	*0.01*	*24.36*	*0.02*	

Improvement of performance by simplification. In most cases simplification increased the numbers of solved problems, while the combination $\mathsf{gwpo}_{01} + \psi$ ties the configuration gwpo_{01} for problems with the negative answer. In particular, as discussed in comparison with MædMax, simplification improves performance for large problems. However, it is to be noted that a configuration without ψ can solve a problem that is not solved by one with ψ. For instance, LCL645-10.001 is solved negatively by gwpo_{01}, but not by $\mathsf{gwpo}_{01}+\psi$. This is due to incompleteness of the heuristics to find a reduction order, as discussed in Section 4.4.

The GWPO. For problems with the negative answer, every problem solved using the GWPO is also solved by the LPO. However, for problems with the positive answer, configurations with the GWPO solve problems that is not solved by the LPO. For instance, both $\mathsf{gwpo}_{01}+\psi$ and gwpo_{01} solve the problem BOO001-1, which is not solved by $\mathsf{lpo} + \psi$ nor lpo. However, configurations with the GWPO quit on a couple of problems due to a memory error, running out of memory.

Chapter 5

Conclusion

In the thesis we presented a new class of non-simplification orders, namely the GWPO, and integrated the ordering into equational theorem proving by maximal ordered completion. We conclude the thesis suggesting two future works.

The GWPO in superposition calculus. Jakubuv and Kaliszyk [16] improved the performance of the E prover [29], which implements superposition calculus [5], by integrating the WPO. They also proposed the relaxed weighted path order (RWPO) as an approximation of the WPO to improve the performance, even though the RWPO is not reduction orders in general. As a consequence, the approach need to bound the number of rewrite steps, as termination is not guaranteed by the ordering. Another potential approach is superposition calculus with the GWPO, and the performance is to be compared. In particular, the performance of superposition calculus on translated problems from higher-order logics of proof assistants is known to be not as desired [7]. This is why new term orderings for superposition calculus are actively studied, such as the recursive path order for applicative terms [10], the transfinite Knuth–Bendix order for applicative terms [7], its extension for combinator equations [9], and the embedding path order [8]. Relationship between these classes and the GWPO is to be investigated in theory and in practice.

Refutational completeness for non-ground-total reduction orders. Refutational completeness is a desired property for theorem proving procedures. If axioms entail a goal, a refutationally complete procedure eventually finds a proof. In particular, refutational completeness of ordered completion [4] and superposition calculus [5] demands ground-total reduction orders. However, the GWPO is not necessarily ground-total. The first step is to investigate a modular proof of completeness for ordered completion [14], and apply obtained techniques to superposition calculus.

Acknowledgment

The thesis is written under the supervision of Hirokawa Nao Sensei. The author is indebted to his indispensable support. Ogawa Mizuhito Sensei supported the author as his second supervisor. Ishihara Hajime Sensei and Yokoyama Keita Sensei supervised his minor research project, which provided a cross-training opportunity. Ishii Daisuke Sensei and Ogata Kazuhiro Sensei gave the author valuable comments at the midterm defense in September 2022. Also, he is thankful to participants in TRS Meetings and group seminars of Hirokawa Laboratory for many stimulating conversations.

References

[1] Release 11.0 · termcomp/tpdb, 2020.

[2] Thomas Aarts and Jurgen Giesl. A collection of examples for termination of term rewriting using dependency pairs. *Technical Report AIB-2001-09*, 2001.

[3] Franz Baader and Tobias Nipkow. *Term rewriting and all that.* Cambridge University Press, 1998.

[4] Leo Bachmair, Nachum Dershowitz, and David A. Plaisted. Completion without failure. In *Rewriting Techniques*, pages 1–30. Elsevier, 1989.

[5] Leo Bachmair and Harald Ganzinger. Rewrite-based equational theorem proving with selection and simplification. *J. Log. Comput.*, 4(3):217–247, 1994.

[6] Clark Barrett, Pascal Fontaine, and Cesare Tinelli. The Satisfiability Modulo Theories Library (SMT-LIB). `www.SMT-LIB.org`, 2016.

[7] Heiko Becker, Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A transfinite Knuth-Bendix order for lambda-free higher-order terms. In Leonardo de Moura, editor, *Automated Deduction - CADE 26 - 26th International Conference on Automated Deduction, Gothenburg, Sweden, August 6-11, 2017, Proceedings*, volume 10395 of *Lecture Notes in Computer Science*, pages 432–453. Springer, 2017.

[8] Alexander Bentkamp. The embedding path order for lambda-free higher-order terms. *FLAP*, 8(10):2447–2470, 2021.

[9] Ahmed Bhayat and Giles Reger. A Knuth-Bendix-like ordering for orienting combinator equations. In Nicolas Peltier and Viorica Sofronie-Stokkermans, editors, *Automated Reasoning - 10th International Joint Conference, IJCAR 2020, Paris, France, July 1-4, 2020, Proceedings, Part I*, volume 12166 of *Lecture Notes in Computer Science*, pages 259–277. Springer, 2020.

[10] Jasmin Christian Blanchette, Uwe Waldmann, and Daniel Wand. A lambda-free higher-order recursive path order. In Javier Esparza and Andrzej S. Murawski, editors, *Foundations of Software Science and Computation Structures - 20th International Conference, FOSSACS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings*, volume 10203 of *Lecture Notes in Computer Science*, pages 461–479, 2017.

[11] Cristina Borralleras, Maria Ferreira, and Albert Rubio. Complete monotonic semantic path orderings. In David A. McAllester, editor, *Automated Deduction - CADE-17, 17th International Conference on Automated Deduction, Pittsburgh, PA, USA, June 17-20, 2000, Proceedings*, volume 1831 of *Lecture Notes in Computer Science*, pages 346–364. Springer, 2000.

[12] Leonardo Mendonça de Moura and Nikolaj S. Bjørner. Z3: an efficient SMT solver. In C. R. Ramakrishnan and Jakob Rehof, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, volume 4963 of *Lecture Notes in Computer Science*, pages 337–340. Springer, 2008.

[13] Nao Hirokawa. Completion and reduction orders (invited talk). In Naoki Kobayashi, editor, *6th International Conference on Formal Structures for Computation and Deduction, FSCD 2021, July 17-24, 2021, Buenos Aires, Argentina (Virtual Conference)*, volume 195 of *LIPIcs*, pages 2:1–2:9. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[14] Nao Hirokawa, Aart Middeldorp, Christian Sternagel, and Sarah Winkler. Abstract completion, formalized. *CoRR*, abs/1802.08437, 2018.

[15] Gérard P. Huet. Confluent reductions: Abstract properties and applications to term rewriting systems: Abstract properties and applications to term rewriting systems. *J. ACM*, 27(4):797–821, 1980.

[16] Jan Jakubuv and Cezary Kaliszyk. Relaxed weighted path order in theorem proving. *Math. Comput. Sci.*, 14(3):657–670, 2020.

[17] Sam Kamin and Jean-Jacques Lévy. Two generalizations of the recursive path ordering. *Unpublished manuscript*, 1980.

[18] Dominik Klein and Nao Hirokawa. Maximal completion. In Manfred Schmidt-Schauß, editor, *Proceedings of the 22nd International Conference on Rewriting Techniques and Applications, RTA 2011, May 30 - June 1, 2011, Novi Sad, Serbia*, volume 10 of *LIPIcs*, pages 71–80. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2011.

[19] Donald E. Knuth and Peter B. Bendix. Simple word problems in universal algebras. In *Automation of Reasoning*, pages 342–376. Springer, 1983.

[20] Martin Korp, Christian Sternagel, Harald Zankl, and Aart Middeldorp. Tyrolean termination tool 2. In Ralf Treinen, editor, *Rewriting Techniques and Applications, 20th International Conference, RTA 2009, Brasília, Brazil, June 29 - July 1, 2009, Proceedings*, volume 5595 of *Lecture Notes in Computer Science*, pages 295–304. Springer, 2009.

[21] Bernd Löchner. Things to know when implementing lpo. *Int. J. Artif. Intell. Tools*, 15(1):53–80, 2006.

[22] Bernd Löchner and Thomas Hillenbrand. A phytography of WALDMEISTER. *AI Commun.*, 15(2-3):127–133, 2002.

[23] Ursula Martin and Tobias Nipkow. Ordered rewriting and confluence. In Mark E. Stickel, editor, *10th International Conference on Automated Deduction, Kaiserslautern, FRG, July 24-27, 1990, Proceedings*, volume 449 of *Lecture Notes in Computer Science*, pages 366–380. Springer, 1990.

[24] Aart Middeldorp, Julian Nagele, and Kiraku Shintani. Confluence competition 2019. In Dirk Beyer, Marieke Huisman, Fabrice Kordon, and Bernhard Steffen, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 25 Years of TACAS: TOOLympics, Held as Part of ETAPS 2019, Prague, Czech Republic, April 6-11, 2019, Proceedings, Part III*, volume 11429 of *Lecture Notes in Computer Science*, pages 25–40. Springer, 2019.

[25] Aart Middeldorp and Hans Zantema. Simple termination of rewrite systems. *Theor. Comput. Sci.*, 175(1):127–158, 1997.

[26] Maxwell Herman Alexander Newman. On theories with a combinatorial definition of" equivalence". *Annals of mathematics*, pages 223–243, 1942.

[27] Robert Nieuwenhuis. Invited talk: Rewrite-based deduction and symbolic constraints. In Harald Ganzinger, editor, *Automated Deduction*

- *CADE-16, 16th International Conference on Automated Deduction, Trento, Italy, July 7-10, 1999, Proceedings*, volume 1632 of *Lecture Notes in Computer Science*, pages 302–313. Springer, 1999.

[28] I. V. Ramakrishnan, R. Sekar, and Andrei Voronkov. Term indexing. In John Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning (in 2 volumes)*, pages 1853–1964. Elsevier and MIT Press, 2001.

[29] Stephan Schulz, Simon Cruanes, and Petar Vukmirovic. Faster, higher, stronger: E 2.3. In Pascal Fontaine, editor, *Automated Deduction - CADE 27 - 27th International Conference on Automated Deduction, Natal, Brazil, August 27-30, 2019, Proceedings*, volume 11716 of *Lecture Notes in Computer Science*, pages 495–507. Springer, 2019.

[30] Nicholas Smallbone. Twee: An equational theorem prover. In André Platzer and Geoff Sutcliffe, editors, *Automated Deduction - CADE 28 - 28th International Conference on Automated Deduction, Virtual Event, July 12-15, 2021, Proceedings*, volume 12699 of *Lecture Notes in Computer Science*, pages 602–613. Springer, 2021.

[31] Geoff Sutcliffe. The CADE ATP system competition - CASC. *AI Mag.*, 37(2):99–101, 2016.

[32] Yoshihito Toyama. Counterexamples to termination for the direct sum of term rewriting systems. *Inf. Process. Lett.*, 25(3):141–143, 1987.

[33] Sarah Winkler and Georg Moser. Mædmax: A maximal ordered completion tool. In Didier Galmiche, Stephan Schulz, and Roberto Sebastiani, editors, *Automated Reasoning - 9th International Joint Conference, IJCAR 2018, Held as Part of the Federated Logic Conference, FloC 2018, Oxford, UK, July 14-17, 2018, Proceedings*, volume 10900 of *Lecture Notes in Computer Science*, pages 472–480. Springer, 2018.

[34] Akihisa Yamada, Keiichirou Kusakari, and Toshiki Sakabe. Unifying the Knuth-Bendix, recursive path and polynomial orders. In Ricardo Peña and Tom Schrijvers, editors, *15th International Symposium on Principles and Practice of Declarative Programming, PPDP '13, Madrid, Spain, September 16-18, 2013*, pages 181–192. ACM, 2013.