PhD Thesis

The Weighted Path Order for Termination of Term Rewriting

April 2014

Akihisa Yamada

Graduate School of Information Science
Nagoya University

Abstract

Termination of *term rewrite systems (TRSs)* has been widely studied and various automated termination provers have been developed. Nevertheless, there is still an increasing demand on the power and scalability of termination provers. This thesis aims at proposing a powerful method for automatically proving termination of TRSs, and moreover providing its efficient implementation. We propose a novel method called the *weighted path order*, which subsumes many existing techniques. In particular, we unify the three most celebrated and longstanding methods: the *Knuth-Bendix order*, the *polynomial interpretation order*, and the *lexicographic path order*. Further to incorporate the new order into the *dependency pair framework*, the modern standard of termination proving, we propose a new technique called *partial status*. In this setting, our method moreover subsumes the matrix interpretation method, as well as many other well-known techniques. Then we present how to encode our method as a *satisfiability modulo theory (SMT)* problem, for which various efficient solvers exist. Our method is implemented as a new termination prover NaTT. With only a few other techniques implemented, NaTT is the second strongest tool in the *International Termination Competition* (full-run 2013), demonstrating the power of our method. We also present new techniques for cooperating with SMT solvers which advance the efficiency of the tool; NaTT ran almost five times faster than any other tool participated in the competition.

In addition, we consider extensions of the Knuth-Bendix order that cope with *associativity and commutativity (AC)* axioms. The orders of Steinbach and of Korovin and Voronkov are revisited; we enhance the former to a more powerful AC-compatible order and modify the latter to amend its lack of monotonicity on non-ground terms. We compare these variants by investigating computational complexity, as well as experiments on problems in termination and completion.

Acknowledgment

Fist of all, I am grateful to my supervisor Toshiki Sakabe for his kind guidance and support for my PhD study. Without him, I could not complete my doctoral research, or probably not even start it. I would like to thank Keiichirou Kusakari for the countless discussions on almost all parts of my study. I would also like to celebrate his recent promotion to a full professor at Gifu University.

I am grateful to Masahiko Sakai for his thoughtful advices that covered not only research topic but also concerning my carrier. I also thank Naoki Nishida for comments from viewpoints of fields that I was unfamiliar with.

It is my great pleasure to have been collaborating with Sarah Winkler, Nao Hirokawa, and Aart Middeldorp. Besides the fact that our joint-work forms a notable chapter in this thesis, I would like to notice that one of the most important ideas of this thesis came into my mind through discussions with them during my comfortable one month stay in Innsbruck.

Finally, I would like to express my gratitude to my friends and family. Although they are not directly related to the topics of this thesis, their existence encouraged me fighting in the challenging situation of a PhD student.

Contents

Chapter 1

Introduction

A *term rewrite system (TRS)* is a computational model that represents each computation step as a rewriting of terms. For example, the function sum that computes summation of list elements can be represented by the following TRS $\mathcal{R}_{\mathsf{sum}}$:

$$\mathcal{R}_{\mathsf{sum}} := \left\{ \begin{array}{rcl} \mathsf{sum}(\mathsf{nil}) & \to & \mathsf{0} \\ \mathsf{sum}(x :: xs) & \to & x + \mathsf{sum}(xs) \end{array} \right.$$

A TRS directly corresponds to a first-order functional program. For example, the TRS $\mathcal{R}_{\mathsf{sum}}$ above corresponds to the following Standard ML program:

```
fun sum nil = 0
  | sum (x :: xs) = x + sum xs
```

By proving *termination* of a TRS one can verify termination of the corresponding first-order functional program. Hence there have been a lot of researches toward automatic termination proving, resulting in many automated termination provers: AProVE [29], $\mathsf{T_TT_2}$ [47], C*i*ME [13], VMTL [67], *etc.*, competing in the annual *termination competition*[1] for a decade.

There is still an increasing demand on power and efficiency of termination provers. Several works have been made to prove termination of more practical programs by transforming them into (constrained) TRSs, e.g. for Haskell [30, 32], Prolog [68], C [21, 22], and Java Bytecode [63]. Moreover, termination of a TRS is a crucial property also for automated equational reasoning, such as the *Knuth-Bendix completion* [43] and *rewriting induction* [64]. In particular, termination provers are incorporated in recent completion tools: Slothrop [78], mkbTT [80, 82],

[1] `http://termination-portal.org/wiki/Termination_Competition`

and KBCV [73]. Hence, this thesis aims at proposing a powerful method of proving termination and providing its efficient implementation.

History of Termination Proving

A *reduction order* is the classic method of proving termination of TRSs. Since the end of 1960s, a number of reduction orders have been proposed. Along with the recent development of *satisfiability (SAT)* solvers and *satisfiability modulo theory (SMT)* solvers, efficient SAT/SMT encodings for these reduction orders have been proposed.

One of the most well-known reduction orders is the *lexicographic path order (LPO)* of Kamin and Lévy [40], a variant of the *recursive path order (RPO)* of Dershowitz [14]. LPO is unified with RPO using *status* [56]. Recently, Codish *et al.* [12] proposed an efficient implementation of RPO with status via SAT encoding.

The *Knuth-Bendix order (KBO)* [43] is the oldest reduction order. KBO has become a practical alternative in automatic termination proving since Korovin and Voronkov [45] discovered a polynomial-time algorithm for proving termination by KBO. Zankl et al. [91] proposed another implementation method via SAT/SMT encoding, and verified a significant improvement in efficiency over dedicated implementations of the polynomial-time algorithm. However, KBO is disadvantageous compared to LPO when *duplicating* rules (where a variable occurs more often in the right-hand side than in the left-hand side) are considered. Actually, no duplicating rule can be oriented by KBO. To overcome this disadvantage, Middeldorp and Zantema [62] proposed the *generalized KBO (GKBO)*, which generalizes weights over algebras that are weakly monotone and *strictly simple*: $f(\dots, x, \dots) > x$. Ludwig and Waldmann proposed another extension of KBO called the *transfinite KBO (TKBO)* [58, 48, 81], which extends the weight function to allow linear polynomials over ordinals. However, proving termination with TKBO involves solving the satisfiability problem of non-linear arithmetic which is undecidable in general. Moreover, TKBO still does not subsume LPO.

The *polynomial order (POLO)* of Lankford [53] interprets each function symbol by a strictly monotone polynomial. Zantema [93] extended the method to algebras, and suggested combining the "max" operator with polynomial interpretations (*max-polynomials* in terms of Fuhs et al. [24]). Fuhs *et al.* proposed an efficient SAT encoding of POLO [23], and a general version of POLO with max [24].

The *dependency pair (DP) method* of Arts and Giesl [2] significantly enhances the classic approach of reduction orders by analyzing cyclic dependencies between rewrite rules. In the DP method, reduction orders are generalized to *reduction pairs*, and it suffices if one rule in a recursive dependency is strictly oriented, and other rules are only weakly oriented. One of the typical methods for designing reduction pairs is *argument filtering* [2], which generates reduction pairs from arbitrary reduction orders. Hence, reduction orders are still an important subject to study in modern termination proving. Another typical technique is generalizing interpretation methods to *weakly* monotone ones, e.g. allowing 0 coefficients for polynomial interpretations [2]. Endrullis et al. [20] extended polynomial interpretations to *matrix interpretations*, and presented their implementation via SAT encoding. More recently, Bofill et al. [8] proposed a reduction pair called *RPOLO*, which unifies standard POLO and RPO by choosing either *RPO-like* or *POLO-like* comparison depending on function symbols.

Overview and Contributions

Chapter 2 presents preliminaries for term rewriting and termination proving, including several existing reduction orders and brief introduction of the dependency pair framework.

Chapter 3 concentrates on generalizations of KBO. Rigorously speaking, GKBO of Middeldorp and Zantema [62] does not generalize KBO, since the strict simplicity is too restrictive. In this chapter we present a slightly modified version of GKBO that subsumes the original KBO. Then a variant introduced by Lankford [55] is revisited, and then related with a more recent proposal by Ludwig and Waldmann [58].

Chapter 4 introduces a new reduction order called the *weighted path order (WPO).* The reduction orders and reduction pairs described so far require different correctness proofs and different implementations. We extract the underlying essence of these techniques and *unify* them into WPO. Technically, WPO is a generalization of GKBO that relaxes the strict simplicity condition of weights to *weak simplicity.* This relaxation becomes possible by combining the recursive checks of LPO with GKBO. While strict simplicity is so restrictive that the original GKBO does not even subsume the standard KBO, weak simplicity is so generous that WPO subsumes not only KBO but also most of the reduction orders described above (LPO, TKBO, POLO and so on), except for matrix interpretations. These results are reported in [85, 87].

Chapter 5 extends WPO as a reduction pair by introducing the notion of *partial status*. This extension further relaxes the weak simplicity condition, and arbitrary weakly monotone interpretations can be used. Hence as a reduction pair, WPO also subsumes the matrix interpretations, as well as KBO, TKBO, LPO and POLO. Though RPOLO also unifies RPO and POLO, we show that WPO and RPOLO are incomparable in general. In practice, WPO shows significant benefit in the problems from the *Termination Problem Data Base (TPDB)* [77], while (the first-order version of) RPOLO does not, as reported in [8]. These results are reported in [87] and partially in [86].

Chapter 6 presents how to encode WPO as an SMT problem, by extending the corresponding technique for KBO proposed by Zankl et al. [91]. In particular, orientability problem of WPO($\mathcal{S}um$), WPO($\mathcal{M}ax$), and WPO($\mathcal{MS}um$) are reduced to a satisfiability problem of linear arithmetic, which is known to be decidable. These results are reported in [85, 87].

Chapter 7 describes the Nagoya Termination Tool (NaTT), which implements WPO as well as some existing techniques of the *DP framework* [36, 27, 29], a successor of DP method. We present some new techniques that advance the efficiency of the tool by deeply cooperating with SMT solvers. Through experiments and results of the *Termination Competition* [75], we also verify the efficiency of our implementation and significance of WPO in practice. The description is reported in [88].

Chapter 8 is devoted to extensions of KBO to cope with *associativity* and *commutativity* (AC) axioms. The orders of Steinbach [70] and of Korovin and Voronkov [46] are revisited; we enhance the former to a more powerful AC-compatible order and modify the latter to amend its lack of monotonicity on non-ground terms. We further present new complexity results. An extension reflecting the proposal of Ludwig and Waldmann [58] in plain KBO is also given. The various orders are compared on problems in termination and completion. These results are reported in [89].

Chapter 2

Preliminaries

In this chapter, we recall some basic notions needed in this thesis: orders, term rewriting, and existing methods for proving termination. In Section 2.1 we present the notions and notations for relations and orders used in this thesis. Section 2.2 recalls basic notions of term rewriting. Classic methods for proving termination based on reduction orders are presented in Section 2.3. Then the more modern dependency pair framework is described in Section 2.4.

2.1 Relations and Orders

A *binary relation* $\sqsupset$ on a set A is a subset of $A \times A$. Following the convention, we write $a \sqsupset b$ instead of $\langle a, b\rangle \in \sqsupset$, and $a_0 \sqsupset_1 a_1 \sqsupset_2 \dots \sqsupset_n a_n$ instead of $a_{i-1} \sqsupset_i a_i$ for every $i \in \{1, \dots, n\}$. Further, we write $a \not\sqsupset b$ to state that $a \sqsupset b$ does not hold.

Definition 2.1 *The* identity *relation on A is denoted by $=_A$. The* composition *$\sqsupset_1 \cdot \sqsupset_2$ of relations $\sqsupset_1$ and $\sqsupset_2$ is defined as: $a \sqsupset_1 \cdot \sqsupset_2 c$ iff $a \sqsupset_1 b \sqsupset_2 c$ for some $b \in A$. From a relation $\sqsupset$ on A, we define the following relations on A:*

- i-folding*: $\sqsupset^0 := =_A$ and $\sqsupset^{i+1} := \sqsupset \cdot \sqsupset^i$,*
- reflexive closure*: $\sqsupset^= := \sqsupset \cup \sqsupset^0$,*
- transitive closure*: $\sqsupset^+ := \bigcup_{i>0} \sqsupset^i$,*
- reflexive transitive closure*: $\sqsupset^* := \bigcup_{i\geq 0} \sqsupset^i$,*
- inverse*: $a \sqsupset^{-1} b$ iff $b \sqsupset a$.*

We denote the inverse of $\sqsupset$ by $\sqsubset$, if such a symbol is available.

Definition 2.2 *A relation $\sqsupset$ on A is said to be*

- reflexive *iff $a \sqsupset a$ for arbitrary $a \in A$,*
- irreflexive *iff $a \not\sqsupset a$ for arbitrary $a \in A$,*
- transitive *iff $a \sqsupset b \sqsupset c$ implies $a \sqsupset c$,*
- *a* quasi-order *iff it is transitive and reflexive,*
- *a* strict order *iff it is transitive and irreflexive,*
- well-founded *iff there exists no infinite sequence $a_1 \sqsupset a_2 \sqsupset \cdots$,*
- total *iff $a \sqsupset b$ or $a = b$ or $a \sqsubset b$ for arbitrary $a, b \in A$.*

In this thesis, we often denote quasi-orders by $\gtrsim$, $\succsim$, *etc.*, and strict orders by $>$, $\succ$, *etc.* Note that $\succ$ does not always denote the *strict part* of $\succsim$, which is defined as $\succsim \setminus \precsim$. Instead, we often assume the following relationship between relations denoted by $\succsim$ and $\succ$:

Definition 2.3 *A relation $\succ$ on a set A is said to be* compatible *with a relation $\succsim$ on A iff $\succsim \cdot \succ \cdot \succsim \subseteq \succ$. An* order pair *on A is a pair $\langle \succsim, \succ \rangle$ of a quasi-order $\succsim$ on A and a strict order $\succ$ which is compatible with $\succsim$. An order pair $\langle \succsim, \succ \rangle$ is said to be* well-founded *iff $\succ$ is well-founded.*

Definition 2.4 *Let $\langle \succsim, \succ \rangle$ be an order pair on a set A. An element $a \in A$ is*

- greatest *(respectively* least*) iff $a \succsim b$ (respectively $a \precsim b$) for every $b \in A$, and*
- maximal *(respectively* minimal*) iff $a \not\prec b$ (respectively $a \not\succ b$) for every $b \in A$.*

A function $f : A^n \to A$ is

- *strictly (weakly)* monotone *in its i-th argument iff*

$$f(a_1, \ldots, a_i, \ldots, a_n) \underset{(\sim)}{\succ} f(a_1, \ldots, a_i', \ldots, a_n)$$

whenever $a_i \underset{(\sim)}{\succ} a_i'$, and

- *strictly (weakly)* simple *in its i-th argument iff $f(a_1, \dots, a_n) \mathrel{{}_{(}\succsim_{)}} a_i$ for every $a_i \in A$.*

We say that f is strictly/weakly monotone/simple iff it is so in every argument.

The *lexicographic* extension is a technique to lift an order pair on a set A to an order pair on finite lists of A. In this thesis, a list of $x_1, \dots, x_n \in A$ is written $[x_1, \dots, x_n]$.

Definition 2.5 (Lexicographic Extension) *Let $\succ$ and $\succsim$ be relations on a set A. The* lexicographic extensions *$\succ^{\mathsf{lex}}$ and $\succsim^{\mathsf{lex}}$ are defined as follows: $[x_1, \dots, x_n] \mathrel{{}_{(}\succsim_{)}^{\mathsf{lex}}} [y_1, \dots, y_m]$ iff for some $k \in \{1, \dots, m\}$ with $n \mathrel{{}_{(}\geq_{)}} k$,*

- *$x_i \succsim y_i$ for each $i \in \{1, \dots, k\}$, and*
- *either $k = m$, or $n > k$ and $x_{k+1} \succ y_{k+1}$.*

Note that the extended relations $\succ^{\mathsf{lex}}$ and $\succsim^{\mathsf{lex}}$ depend on both $\succsim$ and $\succ$.

A similar extension is known for *multisets*: A (finite) *multiset* on A is formally a mapping $X : A \to \mathbb{N}$ such that the set $\{x \mid X(x) > 0\}$ is finite. By $\{x_1, \dots, x_n\}$ we denote the multiset X such that $X(x)$ is the number of indices i satisfying $x_i = x$. Order pairs on A are lifted over multisets as follows:

Definition 2.6 (Multiset Extension) *Let $\succ$ and $\succsim$ be relations on a set A. The* multiset extensions *$\succ^{\mathsf{mul}}$ and $\succsim^{\mathsf{mul}}$ are defined as follows: $X \mathrel{{}_{(}\succsim_{)}^{\mathsf{mul}}} Y$ iff X and Y are written $\{x_1, \dots, x_n\}$ and $\{y_1, \dots, y_m\}$, respectively that satisfy for some $k \in \{1, \dots, m\}$ with $n \mathrel{{}_{(}\geq_{)}} k$,*

- *$x_i \succsim y_i$ for each $i \in \{1, \dots, k\}$, and*
- *for each $j \in \{k+1, \dots, m\}$, there exists $i \in \{k+1, \dots, n\}$ such that $x_i \succ y_j$.*

The following result is folklore; a recent formalization of multiset extensions in Isabelle/HOL is presented in [76].

Theorem 2.7 *If $\langle \succsim, \succ \rangle$ is an order pair, so are $\langle \succsim^{\mathsf{lex}}, \succ^{\mathsf{lex}} \rangle$ and $\langle \succsim^{\mathsf{mul}}, \succ^{\mathsf{mul}} \rangle$.* □

2.2 Terms and Rewriting

Now we introduce the basic notions for terms and term rewriting. For further details of rewriting, see e.g., [3] or [74].

Definition 2.8 (Signature) *A* signature $\mathcal{F}$ *is a finite set of* function symbols*, where each function symbol* f *is associated with its* arity $n \in \mathbb{N}$*. We denote the set of* n*-ary symbols by* $\mathcal{F}_n$*.*

In particular, a function symbol f is called a *constant*, *unary*, or *binary* symbol, if its arity is 0, 1, or 2, respectively.

Definition 2.9 (Term) *Let* $\mathcal{F}$ *be a signature and* $\mathcal{V}$ *a set of* variable symbols*, where* $\mathcal{F} \cap \mathcal{V} = \emptyset$*. The set* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *of* terms *constructed by* $\mathcal{F}$ *and* $\mathcal{V}$ *is inductively defined as follows:*

- $\mathcal{V} \subseteq \mathcal{T}(\mathcal{F}, \mathcal{V})$*, and*
- $f(s_1, \ldots, s_n) \in \mathcal{T}(\mathcal{F}, \mathcal{V})$ *if* $f \in \mathcal{F}_n$ *and* $s_1, \ldots, s_n \in (\mathcal{F}, \mathcal{V})$*.*

The root symbol *of a term* $s = f(s_1, \ldots, s_n)$ *is* f *and denoted by* $\mathsf{root}(s)$*. The set of variables occurring in a term* s *is denoted by* $\mathsf{Var}(s)$*, and the number of occurrence of a variable* x *in* s *is denoted by* $|s|_x$*.*

Throughout this thesis, we consider a fixed set $\mathcal{V}$ of variables. Substitutions to variables are formally defined below.

Definition 2.10 (Substitution) *A mapping* $\theta : \mathcal{V} \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ *is called a substitution if its* domain $\mathsf{Dom}(\theta)$ *is finite, which is defined as follows:*

$$\mathsf{Dom}(\theta) := \{x \in \mathcal{V} \mid \theta(x) \neq x\}$$

A substitution θ *is homomorphically extended to* $\widehat{\theta} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}, \mathcal{V})$ *as follows:*

- $\widehat{\theta}(x) = \theta(x)$ *if* $x \in \mathcal{V}$*, and*
- $\widehat{\theta}(f(s_1, \ldots, s_n)) = f(\widehat{\theta}(s_1), \ldots, \widehat{\theta}(s_n))$*.*

We write $s\theta$ *instead of* $\widehat{\theta}(s)$*.*

Definition 2.11 (Context) *A* context C *is a term in* $\mathcal{T}(\mathcal{F}, \mathcal{V} \cup \{\Box\})$ *such that* $\Box$ *occurs exactly once in* C*. The term obtained by replacing* $\Box$ *in* C *by* s *is denoted by* $C[s]$*.*

Definition 2.12 (Rewrite Relation) *A relation* $\sqsupset$ *on terms is*

- closed under context*, or* monotonic *iff* $s \sqsupset t$ *implies* $C[s] \sqsupset C[t]$ *for every context* C*, and*
- closed under substitution*, or* stable *iff* $s \sqsupset t$ *implies* $s\theta \sqsupset t\theta$ *for every substitution* θ*.*

If a relation on terms is both monotonic and stable, then it is called a rewrite relation*. A rewrite relation is called a* rewrite order *iff it is also a strict order.*

Definition 2.13 (Rewrite Rule) *A* rewrite rule *is a pair of terms* l *and* r*, written* $l \to r$*, such that* $l \notin \mathcal{V}$ *and* $\mathsf{Var}(l) \supseteq \mathsf{Var}(r)$*. A rule* $l \to r$ *is said to be* duplicating *iff there exists a variable* x *such that* $|l|_x < |r|_x$*.*

Definition 2.14 (Term Rewrite System) *A* term rewrite system (TRS) *is a set* $\mathcal{R}$ *of rewrite rules. A TRS* $\mathcal{R}$ *induces the* root reduction $\xrightarrow[\mathcal{R}]{\epsilon}$ *which is defined as follows:* $s \xrightarrow[\mathcal{R}]{\epsilon} t$ *iff* $s = l\theta$ *and* $t = r\theta$ *for some* $l \to r \in \mathcal{R}$ *and substitution* θ*. The* reduction relation $\xrightarrow[\mathcal{R}]{}$ induced by $\mathcal{R}$ *is defined as:* $s \xrightarrow[\mathcal{R}]{} t$ *iff* $s = C[s']$ *and* $t = C[t']$ *such that* $s' \xrightarrow[\mathcal{R}]{\epsilon} t'$*.*

Termination is the main topic of this thesis: A TRS $\mathcal{R}$ is said to be *terminating* iff $\xrightarrow[\mathcal{R}]{}$ is well-founded. In the rest of this chapter we present existing techniques for proving termination of TRSs.

2.3 Reduction Orders

A *reduction order* is a classical technique for proving termination.

Definition 2.15 *A* reduction order *is a well-founded rewrite order. We say an order* $\succ$ orients *a TRS* $\mathcal{R}$ *iff* $l \succ r$ *for every rule* $l \to r \in \mathcal{R}$*; in other words,* $\mathcal{R} \subseteq \succ$*.*

The following result is well-known:

Theorem 2.16 ([54, 92]) *A TRS is terminating iff it is oriented by a reduction order.* $\Box$

To illustrate the technique, we use the following easy TRS as a running example.

Example 2.17 *Consider the following TRS $\mathcal{R}_{\mathsf{fact}}$:*

$$\mathcal{R}_{\mathsf{fact}} := \begin{cases} \mathsf{fact}(\mathsf{0}) & \to & \mathsf{s}(\mathsf{0}) \\ \mathsf{fact}(\mathsf{s}(x)) & \to & \mathsf{s}(x) * \mathsf{fact}(x) \end{cases}$$

In order to prove termination of $\mathcal{R}_{\mathsf{fact}}$ by a reduction order, we need to find a reduction order $\succ$ such that

$$\mathsf{fact}(\mathsf{0}) \succ \mathsf{s}(\mathsf{0})$$
$$\mathsf{fact}(\mathsf{s}(x)) \succ \mathsf{s}(x) * \mathsf{fact}(x)$$

In the rest of this section, we describe existing techniques used to design well-founded reduction orders.

2.3.1 Interpretation Methods

In this section, we investigate methods that ensure well-foundedness by *interpreting* terms into a mathematical structure where a well-founded order is known, e.g., $>$ on the natural numbers $\mathbb{N}$. The idea goes back to Manna and Ness [59].

First, we present in an abstract setting that basically follows Zantema [92] and Endrullis et al. [20].

Definition 2.18 (Algebras) *Let $\mathcal{F}$ be a signature. An $\mathcal{F}$-*algebra *is a pair $\langle A, I\rangle$ that consists of*

- *a set A called the* carrier*, and*
- *a family I of mappings $I(f) : A^n \to A$ called the* interpretation *of $f \in \mathcal{F}_n$.*

A well-founded $\mathcal{F}$-algebra *is a quadruple $\mathcal{A} = \langle A, I, \gtrsim, >\rangle$ such that $\langle A, I\rangle$ is an $\mathcal{F}$-algebra and $\langle \gtrsim, >\rangle$ is a well-founded order pair on A.*

When the signature $\mathcal{F}$ is clear from the context, we simply say algebra instead of $\mathcal{F}$-algebra. For a well-founded algebra $\mathcal{A} = \langle A, I, \gtrsim, >\rangle$, we write $f_{\mathcal{A}}$ instead of $I(f)$ and often omit specifying I. If A is (a subset of) $\mathbb{N}$, then the standard $>$ is assumed and $\gtrsim$ is considered as the standard $\geq$, without explicitly stating. Hence in that case, we only define interpretations $f_{\mathcal{A}}$ in order to define a well-founded algebra $\mathcal{A}$.

Definition 2.19 *Let $\mathcal{A}$ be a well-founded algebra. A mapping $\alpha : \mathcal{V} \to A$ is called an* assignment*, and extended homomorphically to $\widehat{\alpha} : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to A$ as follows:*

- $\widehat{\alpha}(x) = \alpha(x)$ *if $x \in \mathcal{V}$, and*
- $\widehat{\alpha}(f(s_1, \ldots, s_n)) = f_{\mathcal{A}}(\widehat{\alpha}(s_1), \ldots, \widehat{\alpha}(s_n))$.

The relations $\gtrsim_{\mathcal{A}}$ and $>_{\mathcal{A}}$ on terms are defined as follows:

$$s \underset{(\sim)}{>}_{\mathcal{A}} t \overset{\text{def}}{\iff} \widehat{\alpha}(s) \underset{(\sim)}{>} \widehat{\alpha}(t) \text{ for every assignment } \alpha : \mathcal{V} \to A.$$

We extend the terminologies "monotonicity" and "simplicity" on well-founded $\mathcal{F}$-algebras as follows: $\mathcal{A}$ is said to be (strictly/weakly) *monotone* and analogously (strictly/weakly) *simple* iff $f_{\mathcal{A}}$ is so for every $f \in \mathcal{F}$. Further, we just say (strictly/weakly) monotone algebra instead of (strictly/weakly) monotone well-founded $\mathcal{F}$-algebra.

Theorem 2.20 ([92]) *For a strictly monotone algebra $\mathcal{A}$, $>_{\mathcal{A}}$ is a reduction order.* □

Polynomial Interpretations

The *polynomial interpretation order (POLO)* of Lankford [53] is a special case of interpretation methods using a well-founded algebra $\mathcal{P}ol$ such that the carrier set is $\mathbb{N}$ and the interpretation $f_{\mathcal{P}ol}$ is a polynomial for every function symbol $f \in \mathcal{F}$. The algebra $\mathcal{P}ol$ induces a reduction order if it is strictly monotone; in other words, all arguments have coefficients at least 1.

Corollary 2.21 ([53]) *Let $\mathcal{P}ol$ be a strictly monotone polynomial interpretation. Then $>_{\mathcal{P}ol}$ is a reduction order.* □

Example 2.22 *Termination of $\mathcal{R}_{\mathsf{fact}}$ of Example 2.17 can be shown by a polynomial interpretation $\mathcal{P}ol$ such that*

$$\mathsf{s}_{\mathcal{P}ol}(x) = 2x + 1 \qquad \mathsf{0}_{\mathcal{P}ol} = 0$$
$$\mathsf{fact}_{\mathcal{P}ol}(x) = 2x + 2 \qquad x *_{\mathcal{P}ol} y = x + y$$

The left- and right-hand-sides of the rule $\mathsf{fact}(\mathsf{0}) \to \mathsf{s}(\mathsf{0})$ are interpreted as 2 and 1, respectively, and that of the rule $\mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) * \mathsf{fact}(x)$ are $4x + 4$ and $4x + 3$, respectively.

Matrix Interpretations

The *matrix interpretation method* is introduced for string rewriting by Hofbauer and Waldmann [38], and extended for term rewriting by Endrullis et al. [20]. This method uses a well-founded algebra $\mathcal{M}at$ whose carrier is the set $\mathbb{N}^d$ of d-dimension vectors which is ordered by the following order pair $\langle \gtrsim, > \rangle$:

$$\begin{pmatrix} v_1 \\ \vdots \\ v_d \end{pmatrix} \underset{(\sim)}{>} \begin{pmatrix} u_1 \\ \vdots \\ u_d \end{pmatrix} \overset{\text{def}}{\iff} v_1 \underset{(=)}{>} u_1 \text{ and } v_j \geq u_j \text{ for all } j \in \{2, \ldots, d\}$$

The interpretation of $f \in \mathcal{F}_n$ in $\mathcal{M}at$ is in the following shape:

$$f_{\mathcal{M}at}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n) = \boldsymbol{w}_f + \sum_{i=1}^{n} C_{f,i} \cdot \boldsymbol{x}_i$$

where $\boldsymbol{w}_f$ is a d-dimension vector and $C_{f,i}$ is a $d \times d$ matrix for each $i \in \{1, \ldots, n\}$. Variables ranging column vectors are denoted by bold fonts: $\boldsymbol{x}$, $\boldsymbol{y}$, *etc.* The i-th row and j-th column element of a matrix M is denoted by $M^{i,j}$. Monotonicity of $\mathcal{M}at$ is ensured if $C^{1,1}_{f,i} \geq 1$ for every $f \in \mathcal{F}_n$ and $i \in \{1, \ldots, n\}$.

Corollary 2.23 ([20]) *Let $\mathcal{M}at$ be a strictly monotone matrix interpretation. Then $>_{\mathcal{M}at}$ is a reduction order.* □

2.3.2 Recursive Path Orders

In this section, we recall the celebrated technique of Dershowitz [14] for ensuring well-foundedness based on *Kruskal's tree theorem* [49].

Definition 2.24 (Subterm) *A term t is called a* subterm *of a term s, written $s \trianglerighteq t$ iff either $s = t$, or $s = f(s_1, \ldots, s_n)$ and $s_i \trianglerighteq t$ for some $i \in \{1, \ldots, n\}$. A subterm t of s is called a* strict subterm*, written $s \triangleright t$ iff $s \trianglerighteq t$ and $s \neq t$.*

Definition 2.25 (Simplification Order) *A relation $\sqsupset$ on terms is said to have the* subterm property *iff $\triangleright \subseteq \sqsupset$. A* simplification order *is a rewrite order that satisfies the subterm property.*

Theorem 2.26 ([14]) *If the signature $\mathcal{F}$ is finite, then every simplification order is well-founded. Hence, a simplification order is a reduction order.* □

In this thesis, we always consider finite signature. Hence a simplification order is a reduction order. The subterm property can be easily checked:

Proposition 2.27 ([62]) *A rewrite order $\succ$ is a simplification order iff*

$$f(x_1, \ldots, x_n) \succ x_i$$

for every $f \in \mathcal{F}_n$ and $i \in \{1, \ldots, n\}$. $\square$

Multiset Path Order

The *multiset path order* [14] is induced by a *precedence* over function symbols. In this thesis, we consider *quasi-precedences* which are introduced in [40].

Definition 2.28 *A* quasi-precedence $\gtrsim$ *is a quasi-order on $\mathcal{F}$, whose strict part, denoted by $>$, is well-founded. The equivalence part of $\gtrsim$ is denoted by $\sim$.*

Definition 2.29 (MPO) *For a quasi-precedence $\gtrsim$, the* multiset path order (MPO) $>_{\mathsf{MPO}}$ *on terms is recursively defined as follows: $s = f(s_1, \ldots, s_n) >_{\mathsf{MPO}} t$ iff*

(a) $\exists i \in \{1, \ldots, n\}.\ s_i \geq_{\mathsf{MPO}} t$, *or*

(b) $t = g(t_1, \ldots, t_m)$ *and either*

  i. $f > g$ *and* $\forall j \in \{1, \ldots, m\}.\ s >_{\mathsf{MPO}} t_j$, *or*

  ii. $f \sim g$ *and* $\{s_1, \ldots, s_n\} >^{\mathsf{mul}}_{\mathsf{MPO}} \{t_1, \ldots, t_m\}$.

Theorem 2.30 ([14]) *The order $>_{\mathsf{MPO}}$ is a simplification order.* $\square$

Example 2.31 *Termination of $\mathcal{R}_{\mathsf{fact}}$ from Example 2.17 can be shown by MPO. Both rules are oriented by case (b)–i with a precedence such that $\mathsf{fact} > \mathsf{s}$ for the first rule and $\mathsf{fact} > *$ for the second rule.*

Lexicographic Path Order

The *lexicographic path order* of Kamin and Lévy [40] is a variant of MPO that uses lexicographic extension instead of multiset extension when comparing arguments in case (b)–ii. In this thesis, we consider LPO with *status* that allow permutation before the lexicographic comparison (confer [69, 62]).

Definition 2.32 (Status) *A* status function σ *assigns to each function symbol $f \in \mathcal{F}_n$ a permutation $[i_1, \ldots, i_n]$ of positions in $\{1, \ldots, n\}$. We denote the list $[s_{i_1}, \ldots, s_{i_n}]$ by $[s_1, \ldots, s_n]^{\sigma(f)}$ for $\sigma(f) = [i_1, \ldots, i_n]$.*

Definition 2.33 (LPO) *Let $\gtrsim$ be a quasi-precedence and σ a status. The* lexicographic path order (LPO) $>_{\mathsf{LPO}(\sigma)}$ *on terms is recursively defined as follows:* $s = f(s_1, \ldots, s_n) >_{\mathsf{LPO}(\sigma)} t$ *iff*

(a) $\exists i \in \{1, \ldots, n\}.\ s_i \geq_{\mathsf{LPO}(\sigma)} t$, *or*

(b) $t = g(t_1, \ldots, t_m)$, $\forall j \in \{1, \ldots, m\}.\ s >_{\mathsf{LPO}(\sigma)} t_j$ *and either*

  i. $f > g$, *or*

  ii. $f \sim g$ *and* $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{LPO}(\sigma)} [t_1, \ldots, t_m]^{\sigma(g)}$.

When no confusion arises, we write $>_{\mathsf{LPO}}$ instead of $>_{\mathsf{LPO}(\sigma)}$.

Theorem 2.34 ([40]) *The order $>_{\mathsf{LPO}}$ is a simplification order.* □

2.3.3 Knuth-Bendix Order

While being the oldest invented reduction order, the order proposed by Knuth and Bendix [43] (KBO for short) combines both interpretation-based comparison and
precedence-based comparison. The original KBO defines interpretations from a so-called *weight function*.

Definition 2.35 (Weight Function) *A* weight function *is a pair $\langle w, w_0 \rangle$ of mapping $w : \mathcal{F} \to \mathbb{N}$ and $w_0 \in \mathbb{N}$, such that $w(c) \geq w_0$ for every constant $c \in \mathcal{F}_0$. The* weight $w(s)$ *of a term s is defined as follows:*

$$w(s) := \begin{cases} w_0 & \text{if } s \in \mathcal{V} \\ w(f) + \sum_{i=1}^{n} w(s_i) & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

Definition 2.36 (KBO) *Let $\gtrsim$ be a quasi-precedence, $\langle w, w_0 \rangle$ a weight function, and σ a status. The* Knuth-Bendix order (KBO) $>_{\mathsf{KBO}}$ *is recursively defined as follows: $s = f(s_1, \ldots, s_n) >_{\mathsf{KBO}} t$ iff $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either*

1. $w(s) > w(t)$, *or*

2. $w(s) = w(t)$ *and either*

   (a) $t \in \mathcal{V}$, *or*

   (b) $t = g(t_1, \ldots, t_m)$ *and either*

i. $f > g$, *or*

ii. $f \sim g$ *and* $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{KBO}} [t_1, \ldots, t_m]^{\sigma(g)}$.

In order for KBO to be a reduction order, the following condition is required:

Definition 2.37 (Admissibility) *A weight function $\langle w, w_0 \rangle$ is said to be* admissible *for a quasi-precedence $\gtrsim$ iff the following two conditions hold:*

(A) $w_0 > 0$, *and*

(B) *if $f \in \mathcal{F}_1$ and $w(f) = 0$, then f is greatest with respect to $\gtrsim$.*

A detailed proof of the following result can be found in e.g. [3, Theorem 5.4.20].

Theorem 2.38 *Let $\gtrsim$ be a quasi-precedence and $\langle w, w_0 \rangle$ a weight function which is admissible for $\gtrsim$. Then $>_{\mathsf{KBO}}$ is a simplification order.* □

Below we list some remarks on differences between our version and existing formulations of KBO in literatures.

- The original definition by Knuth and Bendix [43] further imposes "$|s|_x = |t|_x$ for all $x \in \mathcal{V}$" in case 2, which was harmlessly removed by Dick et al. [19].

- We do not consider weight functions on real numbers, which was proposed by Martin [61] and adopted in the influential textbook by Baader and Nipkow [3]. According to Korovin and Voronkov [45], however, real numbers do not increase the power of the order when finite TRSs are considered.

- In many literatures [43, 55, 61, 19, 3, 45, 81], case (2a) is phrased as:

  "$s = f^k(t)$ and $t \in \mathcal{V}$ for some $k > 0$,"

  since whenever $w(s) = w_0$, admissibility implies $\mathsf{root}(s) = f$ is unary and has the greatest precedence. As we consider quasi-precedences, such f may be not unique; see a preceding example of Zankl et al. [91]. Use of quasi-precedences in KBO is first mentioned by Dershowitz [14] and defined in [15]; however, the order is defined only on ground terms, and a case corresponding to (2a) of Definition 2.36 does not appear in [15].

- We consider statuses, that are first incorporated in KBO by Steinbach [69]. Our version follows the more general version of Middeldorp and Zantema [62].

The *variable condition* "$|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$" is often said to be a major disadvantage of KBO. Indeed, no duplicating rule can be oriented by KBO.

Example 2.39 *Termination of $\mathcal{R}_{\mathsf{fact}}$ of Example 2.17 cannot be shown by KBO, since x is duplicating in the second rule* $\mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) * \mathsf{fact}(x)$.

2.4 Dependency Pairs

The *dependency pair (DP) method* [2] significantly enhances the classical method of reduction orders by analyzing dependencies between rewrite rules. We briefly recall the essential notions for its successor, the *DP framework* [27, 36, 31].

Definition 2.40 (Defined Symbol) *A function symbol f is* defined *in a TRS $\mathcal{R}$ iff there exists a rule $l \to r \in \mathcal{R}$ such that* $\mathsf{root}(l) = f$. *The set of defined symbols in $\mathcal{R}$ is denoted by $\mathcal{D}$.*

Definition 2.41 (Dependency Pair) *Let $\mathcal{R}$ be a TRS over a signature $\mathcal{F}$. For each $f \in \mathcal{D}$, $\mathcal{F}$ is extended by a fresh* marked symbol *$f^\sharp$ whose arity is the same as f. For $s = f(s_1, \dots, s_n)$ with $f \in \mathcal{D}$, the term $f^\sharp(s_1, \dots, s_n)$ is denoted by $s^\sharp$. The set* $\mathsf{DP}(\mathcal{R})$ *of* dependency pairs *for $\mathcal{R}$ is defined as:*

$$\mathsf{DP}(\mathcal{R}) := \{ l^\sharp \to t^\sharp \mid l \to r \in \mathcal{R}, l \ntriangleright t \trianglelefteq r, \mathsf{root}(t) \in \mathcal{D} \}$$

The exclusion of $l^\sharp \to t^\sharp$ such that $l \triangleright t$ from dependency pairs is due to Dershowitz [16].

Definition 2.42 *A* DP problem *is a pair $\langle \mathcal{P}, \mathcal{R} \rangle$ of a TRS $\mathcal{R}$ and a set $\mathcal{P}$ of dependency pairs for $\mathcal{R}$. A DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ is* finite *iff $\xrightarrow[\mathcal{P}]{\epsilon} \cdot \xrightarrow[\mathcal{R}]{}^*$ is well-founded, where $\mathcal{P}$ is viewed as a TRS.*

The main result of the DP method is the following:

Theorem 2.43 ([2]) *A TRS $\mathcal{R}$ is terminating iff the DP problem $\langle \mathsf{DP}(\mathcal{R}), \mathcal{R} \rangle$ is finite.* □

Finiteness of a DP problem is proved by *DP processors*: A sound *DP processor* gets a DP problem as input and outputs a set of (hopefully simpler) DP problems such that the input problem is finite if all the output problems are finite. Among other DP processors for transforming or simplifying DP problems (confer [31] for a summary), we recall the most important one:

Definition 2.44 (Reduction Pair) *A* reduction pair *is a well-founded order pair $\langle \gtrsim, \succ \rangle$ on terms such that $\gtrsim$ is monotonic and stable, and $\succ$ is stable. A reduction pair $\langle \gtrsim, \succ \rangle$ is* monotonic *iff $\succ$ is monotonic.*

Theorem 2.45 (Reduction Pair Processor) *[2, 27, 36] Let $\langle \gtrsim, \succ \rangle$ be a reduction pair such that $\mathcal{P} \cup \mathcal{R} \subseteq \gtrsim$ and $\mathcal{P}' \subseteq \succ$. Then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound.* □

Example 2.46 *Consider again the TRS $\mathcal{R}_{\mathsf{fact}}$ of Example 2.17. $\mathcal{R}_{\mathsf{fact}}$ has one reduction pair $\mathsf{fact}^\sharp(\mathsf{s}(x)) \to \mathsf{fact}^\sharp(x)$. Hence we need a reduction pair $\langle \gtrsim, \succ \rangle$ that satisfies the following constraints:* [1]

$$\mathsf{fact}(\mathsf{0}) \gtrsim \mathsf{s}(\mathsf{0}) \qquad \mathsf{fact}(\mathsf{s}(x)) \gtrsim \mathsf{s}(x) * \mathsf{fact}(x) \qquad \mathsf{fact}^\sharp(\mathsf{s}(x)) \succ \mathsf{fact}^\sharp(x)$$

2.4.1 Weakly Monotone Interpretations

To define a reduction pair by polynomial interpretations, an interpretation $\mathcal{P}ol$ need not be strictly monotone but only weakly monotone; in other words, 0 coefficients are allowed.

Theorem 2.47 ([2]) *Let $\mathcal{P}ol$ be a weakly monotone polynomial interpretation. Then $\langle \geq_{\mathcal{P}ol}, >_{\mathcal{P}ol} \rangle$ forms a reduction pair.* □

Example 2.48 *Consider an interpretation $\mathcal{P}ol$ such that*

$$\begin{array}{lll} \mathsf{s}_{\mathcal{P}ol}(x) = x + 1 & \mathsf{0}_{\mathcal{P}ol} = 0 & \\ \mathsf{fact}_{\mathcal{P}ol}(x) = x + 1 & x *_{\mathcal{P}ol} y = 0 & \mathsf{fact}^\sharp_{\mathcal{P}ol}(x) = x \end{array}$$

Then $\langle \geq_{\mathcal{P}ol}, >_{\mathcal{P}ol} \rangle$ satisfies all the constraints in Example 2.46.

Theorem 2.49 ([20]) *Let $\mathcal{M}at$ be a matrix interpretation. Then $\langle \gtrsim_{\mathcal{M}at}, >_{\mathcal{M}at} \rangle$ forms a reduction pair.* □

[1]The last constraint can be handled by *subterm criterion* [36], and the first constraints can be ignored if we consider *usable rules* [2]. In order to demonstrate reduction pairs, we ignore these techniques in this chapter.

2.4.2 Argument Filtering

Argument filtering [2, 52] is a typical technique to design a reduction pair from a reduction order.

Definition 2.50 *An* argument filter π *maps each* $f \in \mathcal{F}_n$ *to either a position* $i \in \{1, \ldots, n\}$ *or a list* $[i_1, \ldots, i_m]$ *of positions such that* $1 \leq i_1 < \cdots < i_m \leq n$. *The signature* $\mathcal{F}^\pi$ *consists of every* $f \in \mathcal{F}$ *such that* $\pi(f) = [i_1, \ldots, i_m]$, *and arity of* f *is* m *in* $\mathcal{F}^\pi$. *An argument filter* π *induces a mapping* $\pi : \mathcal{T}(\mathcal{F}, \mathcal{V}) \to \mathcal{T}(\mathcal{F}^\pi, \mathcal{V})$ *as follows:*

$$\pi(s) := \begin{cases} s & \text{if } s \in \mathcal{V} \\ \pi(s_i) & \text{if } s = f(s_1, \ldots, s_n),\ \pi(f) = i \\ f(\pi(s_{i_1}), \ldots, \pi(s_{i_m})) & \text{if } s = f(s_1, \ldots, s_n),\ \pi(f) = [i_1, \ldots, i_m] \end{cases}$$

For an argument filter π *and a reduction order* $\succ$ *on* $\mathcal{T}(\mathcal{F}^\pi, \mathcal{V})$, *the relations* $\succsim^\pi$ *and* $\succ^\pi$ *on* $\mathcal{T}(\mathcal{F}, \mathcal{V})$ *are defined as follows:* $s \underset{(\sim)}{\succ}^\pi t$ *iff* $\pi(s) \underset{(-)}{\succ} \pi(t)$.

Theorem 2.51 ([2]) *For a reduction order* $\succ$ *and an argument filter* π, $\langle \succsim^\pi, \succ^\pi \rangle$ *forms a reduction pair.* □

The effect of argument filtering is especially apparent for KBO; it relaxes the variable condition.

Example 2.52 *Applying an argument filter* π *such that* $\pi(*) = 1$ *for the constraints in Example 2.46, we obtain the following constraints:*

$$\mathsf{fact}(\mathsf{0}) \succsim \mathsf{s}(\mathsf{0}) \qquad \mathsf{fact}(\mathsf{s}(x)) \succsim \mathsf{s}(x) \qquad \mathsf{fact}^\sharp(\mathsf{s}(x)) \succ \mathsf{fact}^\sharp(x)$$

The first constraint can be satisfied by KBO with e.g. $w(\mathsf{fact}) > w(\mathsf{s})$. *The other constraints are satisfied by any instance of KBO.*

Chapter 3

Variants of the Knuth-Bendix Order

In this chapter, we investigate several variants of KBO. Middeldorp and Zantema [62] introduced the *generalized KBO* which, despite the name, does not properly *generalize* KBO. Hence in Section 3.1, we present a slightly modified version that properly subsumes the original KBO. In Section 3.2, we revisit the less-known variant proposed by Lankford [55], and then relate it with the more recent variant proposed by Ludwig and Waldmann [58].

3.1 Generalized Knuth-Bendix Order

Middeldorp and Zantema [62] generalized the weight function of KBO using a weakly monotone algebra which is also strictly simple. However, due to the possibility for unary symbols of weight 0, which is not strictly simple, their order does not rigorously subsume the original KBO. Thus, we present a version that slightly relaxes the condition to admit such unary symbols.

Definition 3.1 (improved GKBO) *Let $\gtrsim$ be a quasi-precedence, σ a status, and $\mathcal{A}$ a well-founded algebra. The* generalized Knuth-Bendix order (GKBO) $>_{\mathsf{GKBO}(\mathcal{A},\sigma)}$ *is recursively defined as follows:* $s = f(s_1, \ldots, s_n) >_{\mathsf{GKBO}(\mathcal{A},\sigma)} t$ *iff*

1. *$s >_{\mathcal{A}} t$, or*

2. *$s \gtrsim_{\mathcal{A}} t$ and either*

   (a) $t \in \mathcal{V}$, or

(b) $t = g(t_1, \ldots, t_m)$ *and either*

i. $f > g$*, or*

ii. $f \sim g$ *and* $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{GKBO}(\mathcal{A},\sigma)} [t_1, \ldots, t_m]^{\sigma(g)}$.

When no confusion arises, we write $>_{\mathsf{GKBO}}$ *or* $>_{\mathsf{GKBO}(\mathcal{A})}$ *instead of* $>_{\mathsf{GKBO}(\mathcal{A},\sigma)}$.

Note that in the above definition, we extend [62] with quasi-order $\gtrsim_{\mathcal{A}}$ and quasi-precedence $\gtrsim$, and omit *multiset status*. However, the most notable difference is that we supply case (2a) which makes the improved GKBO indeed a generalization of the original KBO. By this modification, our version admits unary symbols with greatest precedence to be not strictly simple but only weakly simple.

Definition 3.2 *A well-founded algebra* $\mathcal{A}$ *is* admissible *for a quasi-precedence* $\gtrsim$ *iff for every* $f \in \mathcal{F}$,

- $f_{\mathcal{A}}$ *is strictly simple, or*
- $f_{\mathcal{A}}$ *is weakly simple, and* f *is unary and greatest with respect to* $\gtrsim$.

An equivalent notion of the above admissibility has already been mentioned by Dershowitz [14]. However, since he did not consider variables when defining his variant of KBO, case (2a) does not appear in the paper [14]. For the following result to hold, case (2a) is necessary.

Theorem 3.3 *Let* $\gtrsim$ *be a quasi-precedence and* $\mathcal{A}$ *a weakly monotone algebra which is admissible for* $\gtrsim$*. Then* $>_{\mathsf{GKBO}}$ *is a simplification order.*

Proof. The result follows from the development of Chapter 4 as a corollary of Theorem 4.4 and Theorem 4.6. □

One of the most important advantages of GKBO is that it admits weakly monotone interpretations such as the operator max. Due to this generality, GKBO can orient TRSs that cannot be oriented by KBO.

Example 3.4 *Termination of* $\mathcal{R}_{\mathsf{fact}}$ *of Example 2.17 can be shown by GKBO induced by an algebra* $\mathcal{A}$ *on* $\mathbb{N}$ *with interpretation such that*

$$\mathsf{s}_{\mathcal{A}}(x) = x + 1 \qquad \mathsf{0}_{\mathcal{A}} = 0$$
$$\mathsf{fact}_{\mathcal{A}}(x) = x + 2 \qquad x *_{\mathcal{A}} y = \max\{x, y\} + 1$$

and a precedence such that $\mathsf{fact} > *$*. Note that* $\mathcal{A}$ *is admissible for* $>$*. The first rule* $\mathsf{fact}(\mathsf{0}) \to \mathsf{s}(\mathsf{0})$ *is oriented by case 1. The second rule* $\mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) * \mathsf{fact}(x)$ *is oriented by case (2b)–i, since* $x + 3 \geq \max\{x + 1, x + 2\} + 1$.

Now we prove that the improved GKBO is indeed a generalization of the original KBO. We define a well-founded algebra $\mathcal{S}um$ which plays the role of weights of KBO.

Definition 3.5 *A* summation algebra *is a well-founded algebra $\mathcal{S}um$, whose carrier set is a subset of $\mathbb{N}$ and the interpretation of each $f \in \mathcal{F}$ is in the following form:*

$$f_{\mathcal{S}um}(x_1, \dots, x_n) = w_f + \sum_{i=1}^{n} x_i \tag{3.1}$$

where $w_f \in \mathbb{N}$. We say that $\mathcal{S}um$ corresponds *to a weight function $\langle w, w_0 \rangle$ iff $w_f = w(f)$ for every $f \in \mathcal{F}$ and w_0 is the lower bound of the carrier set.*

Lemma 3.6 *Let $\langle w, w_0 \rangle$ be a weight function and $\mathcal{S}um$ the corresponding summation algebra. Then, $s \mathrel{_{(}\!\geq\!_{)}}_{\mathcal{S}um} t$ iff $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and $w(s) \mathrel{_{(}\!\geq\!_{)}} w(t)$.*

Proof. The "if" direction is trivial. For the "only-if" direction, suppose that $s \mathrel{_{(}\!\geq\!_{)}}_{\mathcal{S}um} t$. Let us define the assignment α_0 which maps all variables to w_0. Then we have $\widehat{\alpha}_0(s) \mathrel{_{(}\!\geq\!_{)}} \widehat{\alpha}_0(t)$, that is, $w(s) \mathrel{_{(}\!\geq\!_{)}} w(t)$. Furthermore, let α_x be the assignment that maps x to $w(s) + w_0$ and other variables to w_0. Then we have $\widehat{\alpha}_x(s) \mathrel{_{(}\!\geq\!_{)}} \widehat{\alpha}_x(t)$, which implies $w(s) + |s|_x \cdot w(s) \mathrel{_{(}\!\geq\!_{)}} w(t) + |t|_x \cdot w(s)$. Here, $|s|_x < |t|_x$ cannot hold since $w(t) \geq 0$. We conclude $|s|_x \geq |t|_x$. □

Note that the above lemma is also a corollary of the *shifting method* of Hong and Jakuš [39, Theorem 1]. Next, we show that admissibility of Definition 2.37 and that of Definition 3.2 corresponds.

Lemma 3.7 *Let $\langle w, w_0 \rangle$ be a weight function, $\mathcal{S}um$ the corresponding summation algebra, and $\gtrsim$ a quasi-precedence. If $\langle w, w_0 \rangle$ is admissible for $\gtrsim$ then $\mathcal{S}um$ is admissible for $\gtrsim$.*

Proof. Consider a function symbol $f \in \mathcal{F}_n$. Since the minimum element of the carrier set is $w_0 > 0$, $f_{\mathcal{S}um}(\dots, x, \dots) > x$ whenever $n > 1$. If $n = 1$, then $f_{\mathcal{S}um}(x) = w(f) + x \not> x$ implies $w(f) = 0$ and hence f has the greatest precedence by the assumption. □

Proposition 3.8 *Let $\langle w, w_0 \rangle$ be a weight function and $\mathcal{S}um$ the corresponding summation algebra. Then, $>_{\mathsf{GKBO}(\mathcal{S}um)} = >_{\mathsf{KBO}}$.*

Proof. Straightforward. □

3.2 Lankford's Polynomial KBO

Lankford [55] proposed a combination of a restricted polynomial interpretation and KBO. In our notation, his result corresponds to the following:

Corollary 3.9 *Let $\mathcal{P}ol$ be a monotone polynomial interpretation. If $\mathcal{P}ol$ is admissible for $\gtrsim$, then $>_{\mathsf{GKBO}(\mathcal{P}ol)}$ is a reduction order.* □

In [55, Lemma 3], he moreover assumed that the lower bound of the carrier set of $\mathcal{P}ol$ is greater than or equal to 2, because he was considering interpretations such as $f_{\mathcal{P}ol}(x, y) = x \cdot y$. We do not explicitly assume this, since when such an interpretation is considered, the admissibility condition demands the lower bound to be above 2.

Example 3.10 *Consider again the TRS $\mathcal{R}_{\mathsf{fact}}$ of Example 2.17. Let $\mathcal{P}ol$ be a polynomial interpretation such that*

$$\mathsf{s}_{\mathcal{P}ol}(x) = 2x \qquad \mathsf{0}_{\mathcal{P}ol} = 1$$
$$\mathsf{fact}_{\mathcal{P}ol}(x) = 2x + 1 \qquad x *_{\mathcal{P}ol} y = x + y$$

The first rule $\mathsf{fact}(\mathsf{0}) \to \mathsf{s}(\mathsf{0})$ is oriented by case 1. Both sides of the second rule $\mathsf{fact}(\mathsf{s}(x)) \to \mathsf{s}(x) * \mathsf{fact}(x)$ are interpreted as $4x + 3$. Hence with precedence $\mathsf{fact} > *$, the rule is oriented by case (2b)–i. Note that the interpretation $*_{\mathcal{P}ol}$ is admissible only if the carrier set does not contain 0.

Note that not all monotone polynomial interpretations can be used in Corollary 3.9, since the admissibility condition imposes every non-unary symbol to be strictly simple. Thus, the interpretation of Example 2.22 cannot be applied. This problem will be overcome in Chapter 4.

3.3 Transfinite KBO

Ludwig and Waldmann [58] proposed an extension of KBO called the *transfinite KBO (TKBO)* that admits weights over *ordinals*, and further introduced *subterm coefficients* when computing weights of terms. However, Kovács et al. [48] showed that weights greater than $\omega^{\omega^{\omega}}$ are not needed. Moreover, when only finite TRSs are considered, Winkler et al. [81] showed that only finite weights are sufficient. Therefore in this thesis, we do not consider transfinite weights.

Definition 3.11 *A* subterm coefficient function *sc assigns a positive integer $sc(f,i)$ to each $f \in \mathcal{F}_n$ and $i \in \{1,\ldots,n\}$. For a weight function $\langle w, w_0\rangle$ and a subterm coefficient function sc, $w(s)$ is refined as follows:*

$$w(s) := \begin{cases} w_0 & \text{if } s \in \mathcal{V} \\ w(f) + \sum_{i=1}^{n} sc(f,i) \cdot w(s_i) & \text{if } s = f(s_1,\ldots,s_n) \end{cases}$$

The variable coefficient $\mathsf{vc}(x,s)$ *of x in s is defined recursively as follows:*

$$\mathsf{vc}(x,s) := \begin{cases} 1 & \text{if } x = s \\ 0 & \text{if } x \neq y \in \mathcal{V} \\ \sum_{i=1}^{n} sc(f,i) \cdot \mathsf{vc}(x,s_i) & \text{if } s = f(s_1,\ldots,s_n) \end{cases}$$

Then the order $>_{\mathsf{TKBO}}$ is obtained by slightly modifying Definition 2.36 using the notions defined above.

Definition 3.12 (TKBO) *Let $\gtrsim$ be a quasi-precedence, $\langle w, w_0\rangle$ a weight function, and sc be a subterm coefficient function. The* transfinite Knuth-Bendix order $>_{\mathsf{TKBO}}$ *with status σ is recursively defined as follows: $s = f(s_1,\ldots,s_n) >_{\mathsf{TKBO}} t$ iff $\mathsf{vc}(x,s) \geq \mathsf{vc}(x,t)$ for all $x \in \mathcal{V}$ and either*

1. *$w(s) > w(t)$, or*
2. *$w(s) = w(t)$ and either*
   (a) *$t \in \mathcal{V}$, or*
   (b) *$t = g(t_1,\ldots,t_m)$ and either*
      i. *$f > g$, or*
      ii. *$f \sim g$ and $[s_1,\ldots,s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{TKBO}} [t_1,\ldots,t_m]^{\sigma(g)}$.*

Theorem 3.13 ([58]) *If $\langle w, w_0\rangle$ is admissible for $\gtrsim$, then $>_{\mathsf{TKBO}}$ is a simplification order.* □

Just as KBO corresponds to GKBO($\mathcal{S}um$), TKBO defined above corresponds to GKBO($\mathcal{P}ol$) where $\mathcal{P}ol$ is a linear polynomial interpretation. Thus it turns out that TKBO is a special case of the polynomial KBO of Lankford [55].

Definition 3.14 *A* linear polynomial interpretation *is a well-founded algebra $\mathcal{P}ol$, whose carrier set is a subset of $\mathbb{N}$ and the interpretation of each $f \in \mathcal{F}$ is in the following form:*

$$f_{\mathcal{P}ol}(x_1, \ldots, x_n) = w_f + \sum_{i=1}^{n} c_{f,i} \cdot x_i \tag{3.2}$$

where $w_f, c_{f,1}, \ldots, c_{f,n} \in \mathbb{N}$. A linear polynomial interpretation $\mathcal{P}ol$ corresponds to *a weight function $\langle w, w_0 \rangle$ and a subterm coefficient function sc iff $w_f = w(f)$, $c_{f,i} = sc(f,i)$, and w_0 is the lower bound of the carrier set.*

Proposition 3.15 *If $\mathcal{P}ol$ corresponds to a weight function $\langle w, w_0 \rangle$ and a subterm coefficient sc, then $>_{\mathsf{GKBO}(\mathcal{P}ol)} = >_{\mathsf{TKBO}}$.*

Proof. Analogous to Proposition 3.8. $\square$

Chapter 4

The Weighted Path Order – as a Reduction Order

In this chapter, we propose a new reduction order called the *weighted path order (WPO)* that further generalizes GKBO by weakening the admissibility condition on algebras. Not only extending GKBO, we show that WPO subsumes most of the reduction orders described so far, except for the matrix interpretations. Moreover, we present examples showing that these subsumption relations are strict.

Instances of WPO are characterized by how weights are computed. In particular, we introduce the following instances of WPO and investigate their relationships with existing reduction orders:

- WPO($\mathcal{S}um$) which uses summations for weight computation. KBO can be obtained as a restricted case of WPO($\mathcal{S}um$), where the *admissibility* condition is enforced on constants and unary symbols. WPO($\mathcal{S}um$) is free from these restrictions, and we verify that each extension strictly increases the power of the order.

- WPO($\mathcal{P}ol$) which uses monotone polynomial interpretations for weight computation. As a reduction order, POLO is subsumed by WPO($\mathcal{P}ol$). TKBO can be obtained as a restricted case of WPO($\mathcal{P}ol$), where interpretations are linear polynomials, admissibility is enforced, and interpretations of constants are greater than 0.

- WPO($\mathcal{M}ax$) which uses maximums for weight computation. LPO can be obtained as a restricted case of WPO($\mathcal{M}ax$), where the weights of all symbols are fixed to 0. In order to keep the presentation simple, we omit *multiset status* and only consider *LPO with status*. Nonetheless, it is easy to extend this result to *RPO with status*.

- WPO($\mathcal{MP}ol$) which combines polynomial and maximum for interpretation, and its variant WPO($\mathcal{MS}um$) whose coefficients are fixed to 1. WPO($\mathcal{MS}um$) generalizes KBO and LPO, and WPO($\mathcal{MP}ol$) moreover subsumes POLO (with max) as a reduction order.

Note that these instances use weakly simple algebras which cannot be used for GKBO in general.

There exist several earlier works on generalizing existing reduction orders. The *semantic path order (SPO)* proposed by Kamin and Lévy [40] is a generalization of RPO where precedence comparison is generalized to an arbitrary well-founded order on terms. However, in order to prove termination by SPO, users have to ensure monotonicity by themselves even if the underlying well-founded order is monotone (confer [9]). On the other hand, monotonicity of WPO is guaranteed. Borralleras et al. [9] proposed a variant of SPO that ensures monotonicity by using an external monotonic order. As well as LPO or POLO, WPO can be used as such an external order. The *general path order (GPO)* [17, 25] is a very general framework that many reduction orders are subsumed. Due to the generality, however, implementing GPO seems to be quite challenging. Indeed, we are not aware of any tool that implements GPO.

In the next section, we give the abstract definition of WPO and prove some properties of the order needed to be a reduction order. Then follows a series of sections that introduce several instances of WPO by fixing algebras. In these sections we investigate relationships between these instances of WPO and existing reduction orders and show the potential of WPO.

4.1 The Definition of Weighted Path Order

We first introduce the definition of WPO. In order to admit algebras that are not strictly but only weakly simple, we employ the recursive checks which ensures that LPO is a simplification order.

Definition 4.1 (WPO) *Let $\gtrsim$ be a quasi-precedence, $\mathcal{A}$ a well-founded algebra and σ a status. The* weighted path order $>_{\mathsf{WPO}(\mathcal{A},\sigma)}$ *on terms is defined as follows:* $s = f(s_1, \ldots, s_n) >_{\mathsf{WPO}(\mathcal{A},\sigma)} t$ *iff*

1. $s >_{\mathcal{A}} t$, or

2. $s \gtrsim_{\mathcal{A}} t$ and

(a) $\exists i \in \{1, \ldots, n\}.\ s_i \geq_{\mathsf{WPO}(\mathcal{A},\sigma)} t$, or

(b) $t = g(t_1, \ldots, t_m)$, $\forall j \in \{1, \ldots, m\}.\ s >_{\mathsf{WPO}(\mathcal{A},\sigma)} t_j$ and either

i. $f > g$ or

ii. $f \sim g$ and $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{WPO}(\mathcal{A},\sigma)} [t_1, \ldots, t_m]^{\sigma(g)}$.

We abbreviate $>_{\mathsf{WPO}(\mathcal{A},\sigma)}$ by $>_{\mathsf{WPO}(\mathcal{A})}$ and $>_{\mathsf{WPO}}$ when no confusion arises.

Case 1 and the side condition of case 2 are the same as GKBO. Case (2a) and the side condition of case (2b) are the recursive checks that correspond to (a) and (b) of LPO. Note that here we may restrict i in (2a) and j in (2b)–ii to positions such that $f(\ldots, x_i, \ldots) \not\gtrsim_{\mathcal{A}} x_i$, since otherwise we have $s >_{\mathcal{A}} t$, which is covered by 1. Cases (2b)–i and (2b)–ii are common among WPO, GKBO and LPO.

Now we show that WPO is a simplification order. Most of the required properties will be obtained in Section 5.1, except for the following:

Lemma 4.2 *If $\mathcal{A}$ is weakly monotone and weakly simple, then $>_{\mathsf{WPO}}$ is monotone.*

Proof. Suppose $s_i >_{\mathsf{WPO}} s'_i$ and let us show

$$s = f(\ldots, s_i, \ldots) >_{\mathsf{WPO}} f(\ldots, s'_i, \ldots) = s'$$

Since $s_i \gtrsim_{\mathcal{A}} s'_i$, we have $s \gtrsim_{\mathcal{A}} s'$ by the weak monotonicity of $\mathcal{A}$. By the weak simplicity of $\mathcal{A}$, we have $s \gtrsim_{\mathcal{A}} s_j$ for every j, and thus $s >_{\mathsf{WPO}} s_j$ by case (2a) of Definition 5.2. Hence case (2b)–ii applies for $s >_{\mathsf{WPO}} s'$. □

Well-foundedness of WPO will be ensured by Lemma 5.8. To meet a theoretical interest, we also state that WPO is a simplification order.

Lemma 4.3 *If $\mathcal{A}$ is weakly simple, then $>_{\mathsf{WPO}}$ has the subterm property.*

Proof. By the assumption, $f(\ldots, s_i, \ldots) \gtrsim_{\mathcal{A}} s_i$ and hence $f(\ldots, s_i, \ldots) >_{\mathsf{WPO}} s_i$ by case (2a). □

Theorem 4.4 *If $\mathcal{A}$ is weakly monotone and weakly simple, then $>_{\mathsf{WPO}}$ is a simplification order.* □

Theorem 4.4 gives an alternative proof for the following result of Zantema [93]:

Theorem 4.5 ([93]) *If a TRS $\mathcal{R}$ is oriented by $>_{\mathcal{A}}$ for a weakly monotone and weakly simple algebra $\mathcal{A}$, then $\mathcal{R}$ is simply terminating, i.e., its termination is shown by a simplification order.*

Proof. Since $>_{\mathcal{A}} \subseteq >_{\mathsf{WPO}}$, $\mathcal{R}$ is oriented by the simplification order $>_{\mathsf{WPO}}$. □

Next, we prove that WPO generalizes our version of GKBO.

Theorem 4.6 *If $\mathcal{A}$ is admissible for $\gtrsim$, then $>_{\mathsf{WPO}} = >_{\mathsf{GKBO}}$.*

Proof. For arbitrary terms $s = f(s_1, \ldots, s_n)$ and t, we show $s >_{\mathsf{WPO}} t$ iff $s >_{\mathsf{GKBO}} t$ by induction on $|s| + |t|$.

- Suppose $s >_{\mathsf{GKBO}} t$. If $s >_{\mathcal{A}} t$, then we have $s >_{\mathsf{WPO}} t$ by 1 of Definition 4.1. Let us consider that $s \gtrsim_{\mathcal{A}} t$ but $s \not>_{\mathcal{A}} t$.
  - Suppose $t \in \mathcal{V}$. Since $s \not>_{\mathcal{A}} t$, the admissibility imposes f to be unary and greatest with respect to $\gtrsim$. If $s_1 \in \mathcal{V}$, then obviously $s_1 = t$ and hence $s = f(t) >_{\mathsf{WPO}} t$ by case (2a). Otherwise, $s_1 >_{\mathsf{GKBO}} t$ by case (2a) of Definition 3.1. By the induction hypothesis, we get $s_1 >_{\mathsf{WPO}} t$. Hence, case (2a) of Definition 4.1 applies for $s >_{\mathsf{WPO}} t$.
  - Suppose $t = g(t_1, \ldots, t_m)$ and case (2b)–i or (2b)–ii applies. For all $j \in \{1, \ldots, m\}$, we have $t >_{\mathsf{GKBO}} t_j$ by the subterm property of $>_{\mathsf{GKBO}}$, and we get $s >_{\mathsf{GKBO}} t_j$ by the transitivity. By the induction hypothesis we obtain $s >_{\mathsf{WPO}} t_j$. Hence the side condition in (2b) of Definition 4.1 is satisfied, and subcase (2b)–i or (2b)–ii applies.
- Suppose $s >_{\mathsf{WPO}} t$. If $s >_{\mathcal{A}} t$, then $s >_{\mathsf{GKBO}} t$ by 1 of Definition 3.1. Consider the case $s \not>_{\mathcal{A}} t$.
  - Suppose $s_i \geq_{\mathsf{WPO}} t$ for some $i \in \{1, \ldots, n\}$. By the induction hypothesis, we have $s_i \geq_{\mathsf{GKBO}} t$. The subterm property of $>_{\mathsf{GKBO}}$ ensures $s >_{\mathsf{GKBO}} s_i$. Hence by the transitivity, we get $s >_{\mathsf{GKBO}} t$.
  - Suppose $t = g(t_1, \ldots, t_m)$. If $f > g$, then case (2b)–i of Definition 3.1 applies. If $f = g$ and $[s_1, \ldots, s_n] >^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]$, then by the induction hypothesis we get $[s_1, \ldots, s_n] >^{\mathsf{lex}}_{\mathsf{GKBO}} [t_1, \ldots, t_m]$, and hence case (2b)–ii applies. □

In the remainder of this section, we investigate several instances of WPO.

4.2 WPO($\mathcal{S}um$)

The first instance of WPO, WPO($\mathcal{S}um$), is induced by a summation algebra $\mathcal{S}um$ introduced by Definition 3.5. Obviously, $\mathcal{S}um$ is strictly (and hence weakly) monotone and weakly simple. We obtain the following as a corollary of Theorem 4.4:

Corollary 4.7 *The relation $>_{\mathsf{WPO}(\mathcal{S}um)}$ is a simplification order.* □

We obtain KBO as a restricted case of WPO($\mathcal{S}um$). The result is a straightforward corollary of Theorem 4.6 and Proposition 3.8.

Corollary 4.8 *Let $\gtrsim$ be a quasi-precedence, $\langle w, w_0 \rangle$ an admissible weight function, and $\mathcal{S}um$ the corresponding summation algebra. Then, $>_{\mathsf{WPO}(\mathcal{S}um)} = >_{\mathsf{KBO}}$.* □

Note that Corollary 4.7 does not impose either condition A or B of Definition 2.37. Let us see that removal of these conditions are indeed advantageous. The following example illustrates that WPO($\mathcal{S}um$) which does not satisfy condition B properly enhances KBO.

Example 4.9 *Consider the following TRS $\mathcal{R}_1$:*

$$\mathcal{R}_1 := \begin{cases} \mathsf{f}(\mathsf{g}(x)) \to \mathsf{g}(\mathsf{f}(\mathsf{f}(x))) \\ \mathsf{f}(\mathsf{h}(x)) \to \mathsf{h}(\mathsf{h}(\mathsf{f}(x))) \end{cases}$$

The first rule is oriented from right to left by LPO in any precedence. The second rule is oriented from right to left by KBO, since $w(\mathsf{h}) > 0$ implies increase in weights and $w(\mathsf{h}) = 0$ implies $\mathsf{h} \gtrsim \mathsf{f}$ by the admissibility. On the other hand, WPO($\mathcal{S}um^+$) with precedence $\mathsf{f} > \mathsf{g}$, $\mathsf{f} > \mathsf{h}$ and $w(\mathsf{g}) > w(\mathsf{f}) = w(\mathsf{h}) = 0$ orients all the rules. Hence, $\mathcal{R}_1$ is orientable by WPO($\mathcal{S}um^+$), but not by KBO or LPO. Note that there is no need to consider a status for $\mathcal{R}_1$, since all symbols are unary.

Moreover, removing condition B is also a proper enhancement. Let us denote WPO($\mathcal{S}um^+$) for WPO($\mathcal{S}um$) if the carrier set of $\mathcal{S}um$ has a positive lower bound.

Example 4.10 *Consider the following TRS $\mathcal{R}_2$:*

$$\mathcal{R}_2 := \begin{cases} \mathsf{f}(\mathsf{a}, \mathsf{b}) \to \mathsf{f}(\mathsf{b}, \mathsf{f}(\mathsf{b}, \mathsf{a})) \\ \mathsf{f}(\mathsf{a}, \mathsf{f}(\mathsf{b}, x)) \to \mathsf{f}(x, \mathsf{f}(\mathsf{b}, \mathsf{b})) \end{cases}$$

The first rule cannot be oriented by KBO or WPO($\mathcal{S}um^+$), since $w(\mathsf{b}) = 0$ is required. The second rule is not orientable by LPO no matter how one chooses σ. On the other hand, WPO($\mathcal{S}um$) with $w(\mathsf{a}) > w(\mathsf{b}) = w(\mathsf{f}) = 0$, $\mathsf{a} > \mathsf{b}$ and $\sigma(f) = [1, 2]$ orients both rules. Hence, $\mathcal{R}_2$ is orientable by WPO($\mathcal{S}um$), but not by LPO, KBO, or WPO($\mathcal{S}um^+$).

4.3 WPO($\mathcal{P}ol$)

Next we consider generalizing WPO($\mathcal{S}um$) using monotone polynomial interpretations. According to Zantema [93, Proposition 4], every monotone interpretation on a totally ordered set is weakly simple. Hence a monotone polynomial interpretation $\mathcal{P}ol$ is weakly simple and we obtain the following:

Corollary 4.11 *If $\mathcal{P}ol$ is strictly monotone then $>_{\mathsf{WPO}(\mathcal{P}ol)}$ is a simplification order.* □

Trivially, POLO is subsumed by WPO($\mathcal{P}ol$) as a reduction order. More precisely, the following relation holds:

Proposition 4.12 *For arbitrary well-founded algebra $\mathcal{A}$, $>_{\mathcal{A}} \subseteq >_{\mathsf{WPO}(\mathcal{A})}$.* □

Also, the following is a direct corollary of Theorem 4.6.

Corollary 4.13 *Let $\mathcal{P}ol$ be a polynomial interpretation and $\gtrsim$ a quasi-precedence. If $\mathcal{P}ol$ is admissible for $\gtrsim$ then $>_{\mathsf{GKBO}(\mathcal{P}ol)} = >_{\mathsf{WPO}(\mathcal{P}ol)}$.* □

Moreover, we can verify that WPO($\mathcal{P}ol$) strictly enhances both POLO and GKBO($\mathcal{P}ol$). More generally, we show that $>_{\mathcal{A}}$ and $>_{\mathsf{GKBO}(\mathcal{A})}$ induced by an interpretation $\mathcal{A}$ on $\mathbb{N}$ cannot subsume even WPO($\mathcal{S}um$).

Example 4.14 *Let $\mathcal{A}$ be a strictly monotone interpretation on $\mathbb{N}$. The first rule of $\mathcal{R}_1$ cannot be oriented by $>_{\mathcal{A}}$:*

$$l_1 = \mathsf{f}(\mathsf{g}(x)) \not>_{\mathcal{A}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x))) = r_1$$

since it is not ω-terminating, according to Zantema [92]. Suppose that $\mathcal{R}_1$ is oriented by $>_{\mathsf{GKBO}(\mathcal{A})}$. For the first rule, we need

$$\mathsf{f}(\mathsf{g}(x)) \geq_{\mathcal{A}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x))) \tag{4.1}$$

We prove $\mathsf{f}(x) =_{\mathcal{A}} x$ *by modifying the proof of [92, Proposition 11].*

First, we show $\mathsf{g}_{\mathcal{A}}(n) \geq n$ *for arbitrary* $n \in \mathbb{N}$*. Suppose on the contrary, there exist* $n \in \mathbb{N}$ *such that* $\mathsf{g}_{\mathcal{A}}(n) < n$*. By monotonicity, we obtain the following infinite reduction sequence:*

$$n > \mathsf{g}_{\mathcal{A}}(n) > \mathsf{g}_{\mathcal{A}}(\mathsf{g}_{\mathcal{A}}(n)) > \dots$$

which contradicts the well-foundedness of $>$ *on* $\mathbb{N}$*. This concludes that* $\mathsf{g}(x) \geq_{\mathcal{A}} x = \mathsf{f}^0(x)$*. Second, suppose that* $\mathsf{g}(x) \geq_{\mathcal{A}} \mathsf{f}^k(x)$*. Then from* (4.1)*,*

$$\mathsf{f}(\mathsf{g}(x)) \geq_{\mathcal{A}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x))) \geq_{\mathcal{A}} \mathsf{f}^k(\mathsf{f}(\mathsf{f}(x))) = \mathsf{f}(\mathsf{f}^{k+1}(x))$$

By monotonicity and totality of $>$ *on* $\mathbb{N}$*, this implies* $\mathsf{g}(x) \geq_{\mathcal{A}} \mathsf{f}^{k+1}(x)$*. Hence, by the induction on* k*, we conclude* $\mathsf{g}(x) \geq_{\mathcal{A}} \mathsf{f}^k(x)$ *for every* $k \in \mathbb{N}$*. If* $\mathsf{f}_{\mathcal{A}}(n) > n$ *for some* $n \in \mathbb{N}$*, then by monotonicity of* $\mathsf{f}_{\mathcal{A}}$ *we obtain the following infinite sequence*

$$n < \mathsf{f}_{\mathcal{A}}(n) < \mathsf{f}_{\mathcal{A}}(\mathsf{f}_{\mathcal{A}}(n)) < \dots$$

which is not possible in order for $\mathsf{g}_{\mathcal{A}}(n) \in \mathbb{N}$ *to be well-defined. We conclude* $\mathsf{f}_{\mathcal{A}}(n) = n$ *for arbitrary* $n \in \mathbb{N}$*, i.e.,* $\mathsf{f}(x) =_{\mathcal{A}} x$*.*

Now we are ready to consider the second rule of $\mathcal{R}_1$*:*

$$l_2 = \mathsf{f}(\mathsf{h}(x)) \to \mathsf{h}(\mathsf{h}(\mathsf{f}(x))) = r_2$$

Since $\mathsf{f}(x) =_{\mathcal{A}} x$*, we need* $\mathsf{h}(x) \geq_{\mathcal{A}} \mathsf{h}(\mathsf{h}(x))$*. This implies* $\mathsf{h}_{\mathcal{A}}(n) \not> n$*. Now the admissibility demands both* f *and* h *to be the greatest with respect to the precedence* $\gtrsim$*, and thus* $\mathsf{f} \sim \mathsf{h}$*. Hence this rule can be oriented only from right to left by GKBO(*$\mathcal{A}$*).*

Note also that it is not possible to orient one of the rules in $\mathcal{R}_1$ *by* $>_{\mathcal{P}ol}$ *and the other by* $\geq_{\mathcal{P}ol}$*. Thus, togather with the discussion in Example 4.9, we conclude that* $\mathcal{R}_1$ *cannot be oriented by any lexicographic composition of POLO, (T)KBO, and LPO.* □

4.4 WPO($\mathcal{M}ax$)

The algebras considered in the preceding sections are strictly monotone. WPO admits also *weakly* monotone interpretations; a typical example is the operator max. This section considers instances of WPO using max for interpretation. Note that a weakly monotone polynomial (i.e., 0 coefficients are allowed) is not weakly simple, and hence cannot be applied for WPO of this chapter. We will consider such polynomials in Chapter 5.

Definition 4.15 *A* maximum algebra $\mathcal{M}ax$ *is a well-founded algebra whose carrier set is* $\mathbb{N}$ *and the interpretation of* $f \in \mathcal{F}_n$ *is of the following form:*

$$\begin{aligned} f_{\mathcal{M}ax} &:= w_f && \text{if } n = 0 \\ f_{\mathcal{M}ax}(x_1, \dots, x_n) &:= \max_{i=1}^{n} \left(p_{f,i} + x_i\right) && \text{otherwise} \end{aligned}$$

Lemma 4.16 *The algebra* $\mathcal{M}ax$ *is weakly monotone and weakly simple.*

Proof. Weak simplicity is obvious from the fact that $\max(\dots, a, \dots) \geq a$. For weak monotonicity, suppose $a > b$ and let us show

$$a' = f_{\mathcal{M}ax}(c_1, \dots, c_k, a, d_1, \dots, d_l) \geq f_{\mathcal{M}ax}(c_1, \dots, c_k, b, d_1, \dots, d_l) = b'$$

To this end, let $c = f_{\mathcal{M}ax}(c_1, \dots, c_k, 0, d_1, \dots, d_l)$. If $c \geq p_{f,k+1} + a$, then $a' = b' = c$. Otherwise, we have $a' = p_{f,k+1} + a$ and either $a' > p_{f,k+1} + b = b' > c$ or $a' > b' = c$. □

Note that $\mathcal{M}ax$ can be considered as the 1-dimensional variant of *arctic interpretations* [44]. The weak monotonicity of $\mathcal{M}ax$ is also shown there. From the above lemma, we obtain the following corollary of Theorem 4.4:

Corollary 4.17 *The relation* $>_{\mathsf{WPO}(\mathcal{M}ax)}$ *is a simplification order.* □

Now we show that LPO is obtained as a restricted case of WPO($\mathcal{M}ax$).

Theorem 4.18 *Let* $\mathcal{M}ax$ *be a maximum algebra such that for every* $f \in \mathcal{F}_n$, $w_f = 0$ *and* $p_{f,i} = 0$ *for every* $i \in \{1, \dots, n\}$. *Then,* $>_{\mathsf{LPO}} = >_{\mathsf{WPO}(\mathcal{M}ax)}$.

Proof. From the assumptions, $s >_{\mathcal{M}ax} t$ never holds. Hence, case 1 of Definition 4.1 can be ignored. Moreover, $s \geq_{\mathcal{M}ax} t$ is equivalent to $\mathsf{Var}(s) \supseteq \mathsf{Var}(t)$. One can easily verify the latter holds whenever $s >_{\mathsf{LPO}} t$, using the fact that $s \not\geq_{\mathsf{LPO}} x$ for $x \notin \mathsf{Var}(s)$. Hence, the side condition of case 2 can be ignored and Definition 2.33 and Definition 4.1 become equivalent. □

The following example illustrates that WPO($\mathcal{M}ax$) properly enhances LPO.

Example 4.19 *Consider the following TRS* $\mathcal{R}_3$*:*

$$\mathcal{R}_3 := \begin{cases} \mathsf{f}(x, y) \to \mathsf{g}(x) \\ \mathsf{f}(\mathsf{g}(x), y) \to \mathsf{f}(x, \mathsf{g}(x)) \\ \mathsf{f}(x, \mathsf{g}(y)) \to \mathsf{f}(y, y) \end{cases}$$

To orient the first two rules by LPO, we need $\mathsf{f} > \mathsf{g}$ *and* $\sigma(\mathsf{f}) = [1, 2]$. *LPO cannot orient the third rule by this precedence and status, while* $p_{\mathsf{g},1} > p_{\mathsf{f},1} = 0$ *suffices for WPO(*$\mathcal{M}ax$*). Since the last two rules are duplicating, KBO or WPO(*$\mathcal{S}um$*) cannot apply for* $\mathcal{R}_3$.

4.5 WPO($\mathcal{MP}ol$) and WPO($\mathcal{MS}um$)

Note that WPO($\mathcal{M}ax$) does not cover WPO($\mathcal{S}um$), and not even KBO. Hence in this section, we consider unifying WPO($\mathcal{M}ax$) and WPO($\mathcal{P}ol$) to cover both KBO and LPO. To this end, we introduce the *weight status* to choose a polynomial or max for each function symbol.

Definition 4.20 *A* weight status function *is a mapping ws which maps each function symbol f either to symbol* pol *or to* max. *The well-founded algebra $\mathcal{MP}ol$ consists of the carrier set $\{a \in \mathbb{N} \mid a \geq w_0\}$ and the interpretation which is defined as follows:*

$$f_{\mathcal{MP}ol}(a_1, \ldots, a_n) := \begin{cases} w_f + \sum_{i=1}^{n} c_{f,i} \cdot a_i & \text{if } ws(f) = \mathsf{pol} \\ \max_{i=1}^{n}(p_{f,i} + a_i) & \text{if } ws(f) = \mathsf{max} \end{cases}$$

We denote $\mathcal{MP}ol$ by $\mathcal{MS}um$ if coefficients are at most 1.

Corollary 4.21 *The relation $>_{\mathsf{WPO}(\mathcal{MP}ol)}$ is a simplification order.* □

Trivially, WPO($\mathcal{MS}um$) subsumes both WPO($\mathcal{S}um$) and WPO($\mathcal{M}ax$). Hence, we obtain the following more influential result:

Theorem 4.22 *WPO($\mathcal{MS}um$) subsumes both LPO and KBO.* □

As far as we know, this is the first simplification order that unifies LPO and KBO. The following example illustrates that WPO($\mathcal{MS}um$) is strictly stronger than the union of WPO($\mathcal{S}um$) and WPO($\mathcal{M}ax$).

Example 4.23 *Consider the following TRS $\mathcal{R}_4$:*

$$\mathcal{R}_4 := \begin{cases} \mathsf{f}(\mathsf{f}(x, y), z) \to \mathsf{f}(x, \mathsf{f}(y, z)) \\ \mathsf{g}(\mathsf{f}(\mathsf{a}, x), \mathsf{b}) \to \mathsf{g}(\mathsf{f}(x, \mathsf{b}), x) \end{cases}$$

If $ws(\mathsf{f}) = \mathsf{max}$, then the first rule requires $p_{\mathsf{f},2} = 0$. Under this restriction the second rule cannot be oriented. If $ws(\mathsf{f}) = \mathsf{pol}$, then the first rule is oriented iff $c_{f,1} = c_{f,2} = 1$ and $\sigma(f) = [1, 2]$. On the other hand, the duplicating variable x in the second rule requires $ws(\mathsf{g}) = \mathsf{max}$. Hence, $\mathcal{R}_4$ is orientable by WPO($\mathcal{MS}um$) only if $ws(\mathsf{f}) = \mathsf{pol}$ and $ws(\mathsf{g}) = \mathsf{max}$.

The following example suggests that WPO($\mathcal{MS}um$) advances the state-of-the-art of automated termination proving.

Example 4.24 *The most powerful termination provers including* APro<span>VE 2013</span> *and* $\mathsf{T_TT_2}$ *1.11 fail to prove termination of the following TRS* $\mathcal{R}_5$*:*

$$\mathcal{R}_5 := \begin{cases} \mathsf{f}(\mathsf{g}(\mathsf{g}(x,\mathsf{a}),\mathsf{g}(\mathsf{b},y))) \to \mathsf{f}(\mathsf{g}(\mathsf{g}(\mathsf{h}(x,x),\mathsf{b}),\mathsf{g}(y,\mathsf{a}))) \\ \mathsf{g}(x,y) \to x \\ \mathsf{h}(x,\mathsf{h}(y,z)) \to y \end{cases}$$

We show that WPO($\mathcal{MS}um$*) such that*

$$\mathsf{a}_{\mathcal{MS}um} = 1 \qquad \mathsf{g}_{\mathcal{MS}um}(x,y) = x + y$$
$$\mathsf{b}_{\mathcal{MS}um} = 0 \qquad \mathsf{h}_{\mathcal{MS}um}(x,y) = \max\{x,y\}$$

and $\sigma(\mathsf{f}) = \sigma(\mathsf{g}) = [1,2]$ *orients all the rules. For the first rule, applying case (2b)–ii twice it yields orienting* $\mathsf{g}(x,\mathsf{a}) >_{\mathsf{WPO}(\mathcal{MS}um)} \mathsf{g}(\mathsf{h}(x,x),\mathsf{b})$ *where case 1 applies. The other rules are trivially oriented.*

In general, matrix interpretations cannot be combined with WPO, since a matrix interpretation is often not weakly simple. Consider the following TRS from [20]:

$$\mathcal{R}_6 := \{\mathsf{f}(\mathsf{f}(x)) \to \mathsf{f}(\mathsf{g}(\mathsf{f}(x)))\}$$

which is shown terminating by the following matrix interpretation $\mathcal{M}at$ such that

$$\mathsf{f}_{\mathcal{M}at}(\vec{x}) = \begin{pmatrix}1 & 1\\ 0 & 0\end{pmatrix} \cdot \vec{x} + \begin{pmatrix}0\\1\end{pmatrix} \qquad \mathsf{g}_{\mathcal{M}at}(\vec{x}) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} \cdot \vec{x}$$

However, $\mathsf{g}_{\mathcal{M}at}$ is not weakly simple. For example,

$$\mathsf{g}_{\mathcal{M}at}(\begin{pmatrix}0\\1\end{pmatrix}) = \begin{pmatrix}1 & 0\\ 0 & 0\end{pmatrix} \cdot \begin{pmatrix}0\\1\end{pmatrix} = \begin{pmatrix}0\\0\end{pmatrix} \not\geq \begin{pmatrix}0\\1\end{pmatrix}$$

Hence, in order to unify the matrix interpretation with WPO, we have to further relax the weak simplicity condition on interpretations. This will be achieved in the next chapter by extending WPO to an reduction pair.

Chapter 5

WPO as a Reduction Pair

In modern termination provers, reduction orders are often incorporated in the DP framework as reduction pairs. Since WPO introduced in Chapter 4 is a reduction order, it is possible to obtain a reduction pair using argument filtering [2]. However, in this approach WPO cannot subsume reduction pairs induced by weakly monotone interpretations, including POLO. This is because the weak part of POLO is not subsumed by the weak part of WPO, i.e., $\geq^{\pi}_{\mathsf{WPO}(\mathcal{A})}$. Hence in this chapter, we propose the notion of a *partial status* that turns WPO into a reduction pair. This extension further relaxes the weak simplicity condition, and arbitrary weakly monotone interpretations can be used for weight computation. Hence as a reduction pair, WPO also subsumes the matrix interpretations, as well as GKBO, LPO, and POLO. Though RPOLO also unifies RPO and POLO, we show that WPO and RPOLO are incomparable in general. In practice, WPO brings significant benefit on the problems from the *Termination Problem Data Base (TPDB)* [77], while (the first-order version of) RPOLO does not, as reported in [8].[1]

5.1 WPO with Partial Status

In this section, we introduce a reduction pair by incorporating a *partial status* into WPO. A partial status is a generalization of status that admits *non-permutations.*

[1]Note that in theory, it is possible to consider an instance of WPO that uses RPOLO for weight computation.

Definition 5.1 *A* partial status function σ *is a mapping that assigns to each n-ary symbol f a list $[i_1, \ldots, i_m]$ of (distinct) positions in $\{1, \ldots, n\}$. We also view $\sigma(f)$ as the set $\{i_1, \ldots, i_m\}$ for $\sigma(f) = [i_1, \ldots, i_m]$.*

Note that here $\sigma(f)$ need not be a permutation, since some positions may be ignored. If every $\sigma(f)$ is a permutation, then we say that σ is *total.* Conversely if $\sigma(f) = [\,]$ for every f, then we call σ the *empty* status.

Definition 5.2 (WPO with Partial Status) *Let $\mathcal{A}$ be a well-founded algebra and σ a partial status. The relations $\gtrsim_{\mathsf{WPO}(\mathcal{A},\sigma)}$ and $>_{\mathsf{WPO}(\mathcal{A},\sigma)}$ on terms are defined by mutual recursion as follows: For $s \in \mathcal{V}$, $s \gtrsim_{\mathsf{WPO}(\mathcal{A},\sigma)} t$ iff $s = t$. For $s = f(s_1, \ldots, s_n)$, $s \mathrel{\overset{>}{(\sim)}}_{\mathsf{WPO}(\mathcal{A},\sigma)} t$ iff*

1. *$s >_{\mathcal{A}} t$, or*
2. *$s \gtrsim_{\mathcal{A}} t$ and*
   - *(a) $\exists i \in \sigma(f).\ s_i \gtrsim_{\mathsf{WPO}(\mathcal{A},\sigma)} t$, or*
   - *(b) $t = g(t_1, \ldots, t_m)$, $\forall j \in \sigma(g).\ s >_{\mathsf{WPO}(\mathcal{A},\sigma)} t_j$ and either*
     - *i. $f > g$ or*
     - *ii. $f \sim g$ and $[s_1, \ldots, s_n]^{\sigma(f)} \mathrel{\overset{>}{(\sim)}}{}^{\mathsf{lex}}_{\mathsf{WPO}(\mathcal{A},\sigma)} [t_1, \ldots, t_m]^{\sigma(g)}$.*

The advantage of a partial status over argument filtering is due to the *weights* of ignored arguments. This is illustrated by the following example.

Example 5.3 *Consider a DP problem that induces the following constrains:*

$$\mathsf{f}^\sharp(\mathsf{s}(x)) \succ \mathsf{f}^\sharp(\mathsf{p}(\mathsf{s}(x))) \qquad \mathsf{p}(\mathsf{s}(x)) \succsim x$$

In order to satisfy the first constraint by any simplification order, the argument of p must be filtered. However, the second constraint cannot be satisfied under such an argument filtering.

On the other hand, the DP problem can be shown finite using WPO with partial status such that $\mathsf{s}_{\mathcal{A}}(x) > x$, $\mathsf{f}^\sharp_{\mathcal{A}}(x) = \mathsf{p}_{\mathcal{A}}(x) = x$, $\sigma(\mathsf{f}^\sharp) = \sigma(\mathsf{s}) = [1]$, $\sigma(\mathsf{p}) = [\,]$, and $\mathsf{s} > \mathsf{p}$. We have $\mathsf{f}^\sharp(\mathsf{s}(x)) >_{\mathsf{WPO}} \mathsf{f}^\sharp(\mathsf{p}(\mathsf{s}(x)))$ because of cases (2b)–ii and (2b)–i, and $\mathsf{p}(\mathsf{s}(x)) \gtrsim_{\mathsf{WPO}} x$ because of case 1.

In order for Definition 5.2 to define a correct reduction pair, even weak simplicity of $\mathcal{A}$ can be relaxed to the following:

Definition 5.4 *A well-founded algebra $\mathcal{A}$ is* (weakly) simple *with respect to σ iff $f_{\mathcal{A}}$ is (weakly) simple in its i-th argument for every $f \in \mathcal{F}$ and $i \in \sigma(f)$.*

In the rest of this section, we prove that WPO with partial status forms a reduction pair. The following auxiliary result is obvious from the definition.

Lemma 5.5 *The following inclusion holds: $>_{\mathsf{WPO}} \subseteq \gtrsim_{\mathsf{WPO}}$.* □

Using the above lemma, we show compatibility of $\gtrsim_{\mathsf{WPO}}$ and $>_{\mathsf{WPO}}$.

Lemma 5.6 (Compatibility) *The relation $>_{\mathsf{WPO}}$ is compatible with $\gtrsim_{\mathsf{WPO}}$.*

Proof. Supposing $s \gtrsim_{\mathsf{WPO}} t >_{\mathsf{WPO}} u$, we show $s >_{\mathsf{WPO}} u$ by induction on $\langle |s|, |t|, |u| \rangle$. The other case, $s >_{\mathsf{WPO}} t \gtrsim_{\mathsf{WPO}} u$, is analogous.

From the definition, it is obvious that s and t are in form $f(s_1, \ldots, s_n)$ and $g(t_1, \ldots, t_m)$, respectively, and moreover $s \gtrsim_{\mathcal{A}} t \gtrsim_{\mathcal{A}} u$. If $s >_{\mathcal{A}} t$ or $t >_{\mathcal{A}} u$, then we obtain $s >_{\mathsf{WPO}} u$ by case 1. If $s_i \gtrsim_{\mathsf{WPO}} t$ for some $i \in \sigma(f)$, then by the induction hypothesis and Lemma 5.5, we get $s_i \gtrsim_{\mathsf{WPO}} u$ and hence case (2a) applies for $s >_{\mathsf{WPO}} t$. Now suppose $s >_{\mathsf{WPO}} t_j$ for every $j \in \sigma(g)$ and either $f > g$ or $f \sim g$ and $[s_1, \ldots, s_n]^{\sigma(f)} \gtrsim^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]^{\sigma(g)}$ holds. There remain the following cases to consider for $t >_{\mathsf{WPO}} u$:

- $t_j \gtrsim_{\mathsf{WPO}} u$ for some $j \in \sigma(g)$. In this case, we have $s >_{\mathsf{WPO}} t_j \gtrsim_{\mathsf{WPO}} u$. Hence the induction hypothesis concludes $s >_{\mathsf{WPO}} u$.

- $u = h(u_1, \ldots, u_l)$ and $t >_{\mathsf{WPO}} u_k$ for every $k \in \sigma(h)$. By the induction hypothesis, we obtain $s >_{\mathsf{WPO}} u_k$. Moreover, if $f > g$ or $g > h$, then we get $f > h$ and (2b)–i applies. Otherwise, we have $f \sim g \sim h$ and $[s_1, \ldots, s_n]^{\sigma(f)} \gtrsim^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]^{\sigma(g)} >^{\mathsf{lex}}_{\mathsf{WPO}} [u_1, \ldots, u_k]^{\sigma(h)}$. From the induction hypothesis, we obtain $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{WPO}} [u_1, \ldots, u_k]^{\sigma(h)}$. Hence (2b)–ii applies. □

In order to prove well-foundedness of $>_{\mathsf{WPO}}$, we define the set SN of *strongly normalizing* terms as follows: $s \in \mathsf{SN}$ iff there exist no infinite reduction sequence $s >_{\mathsf{WPO}} s_1 >_{\mathsf{WPO}} s_2 >_{\mathsf{WPO}} \ldots$ beginning from s. In the next two lemmas, we prove that all terms are in SN.

Lemma 5.7 *Suppose that $\mathcal{A}$ is weakly simple with respect to σ. If $s = f(s_1, \ldots, s_n)$ with $s_i \in \mathsf{SN}$ for every $i \in \sigma(f)$, then $s >_{\mathsf{WPO}} t$ implies $t \in \mathsf{SN}$.*

Proof. We perform induction on $\langle s, f, [s_1, \ldots, s_n]^{\sigma(f)}, |t| \rangle$ which is ordered by the lexicographic composition of $>_{\mathcal{A}}$, $>$, $>^{\mathsf{lex}}_{\mathsf{WPO}}$ and $>$. Since it is obvious if $t \in \mathsf{Var}$, we consider $t = g(t_1, \ldots, t_m)$.

1. Suppose $s >_{\mathcal{A}} t$. First we show that $t_j \in \mathsf{SN}$ for every $j \in \sigma(g)$. By the weak simplicity assumption, we have $s >_{\mathcal{A}} t \gtrsim_{\mathcal{A}} t_j$ and hence $s >_{\mathsf{WPO}} t_j$. Thus by the induction hypothesis on the fourth component, we obtain $t_j \in \mathsf{SN}$. Now for arbitrary u such that $t >_{\mathsf{WPO}} u$, the induction hypothesis on the first component yields $u \in \mathsf{SN}$.

2. Suppose $s \gtrsim_{\mathcal{A}} t$. There are two subcases to consider.

   (a) Suppose $s_i \gtrsim_{\mathsf{WPO}} t$ for some $i \in \sigma(f)$. Then by the assumption, we have $s_i \in \mathsf{SN}$. Hence by Lemma 5.6, we obtain $t \in \mathsf{SN}$.

   (b) Suppose $s >_{\mathsf{WPO}} t_j$ for every $j \in \sigma(g)$. Then by the induction hypothesis on the fourth component, we get $t_j \in \mathsf{SN}$. Consider arbitrary u such that $t >_{\mathsf{WPO}} u$. Since we have either $f > g$ or $f \gtrsim g$ and $[s_1, \ldots, s_n]^{\sigma(f)} >^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]^{\sigma(g)}$, $\langle s, f, [s_1, \ldots, s_n]^{\sigma(f)}, |t| \rangle$ is greater than $\langle t, g, [t_1, \ldots, t_m]^{\sigma(g)}, |u| \rangle$ by second or third components. Hence, the induction hypothesis yields $u \in \mathsf{SN}$. □

Lemma 5.8 (Well-foundedness) *If $\mathcal{A}$ is weakly simple with respect to σ, then $>_{\mathsf{WPO}}$ is well-founded.*

Proof. Let us show $s \in \mathsf{SN}$ for every term s by induction on $|s|$. It is trivial if $s \in \mathcal{V}$. Suppose $s = f(s_1, \ldots, s_n) >_{\mathsf{WPO}} t$. By the induction hypothesis, we have $s_i \in \mathsf{SN}$ for every $i \in \{1, \ldots, n\}$. Hence by Lemma 5.7, we get $t \in \mathsf{SN}$. □

Now we prove that $\gtrsim_{\mathsf{WPO}}$ and $>_{\mathsf{WPO}}$ are quasi- and strict orders, respectively.

Lemma 5.9 (Transitivity) *Both $\gtrsim_{\mathsf{WPO}}$ and $>_{\mathsf{WPO}}$ are transitive.*

Proof. Analogous to Lemma 5.6. □

Lemma 5.10 ((Ir)reflexivity) *$\gtrsim_{\mathsf{WPO}}$ is reflexive and $>_{\mathsf{WPO}}$ is irreflexive.*

Proof. For arbitrary term s, $s \gtrsim_{\mathsf{WPO}} s$ is easy by induction on $|s|$. Irreflexivity of $>_{\mathsf{WPO}}$ follows from Lemma 5.8. □

Now remaining properties required for a reduction pair are *stability* and *weak monotonicity*, which are shown bellow.

Lemma 5.11 (Stability) *Both $\gtrsim_{\mathsf{WPO}}$ and $>_{\mathsf{WPO}}$ are stable.*

Proof. Let $s = f(s_1, \dots, s_n) \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}} t$ and θ an arbitrary substitution. We show $s\theta \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}} t\theta$ by induction on $|s| + |t|$. It is obvious if $s >_{\mathcal{A}} t$. Otherwise, we have $s \gtrsim_{\mathcal{A}} t$ and obviously $s\theta \gtrsim_{\mathcal{A}} t\theta$. The remaining cases are as follows:

- Suppose $s_i \gtrsim_{\mathsf{WPO}} t$ for some $i \in \sigma(f)$. By the induction hypothesis, we get $s_i\theta \gtrsim_{\mathsf{WPO}} t\theta$. Hence, case (2a) applies for $s\theta \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}} t\theta$.

- Suppose $t = g(t_1, \dots, t_m)$ and $s >_{\mathsf{WPO}} t_j$ for all $j \in \sigma(g)$. By the induction hypothesis, we get $s\theta >_{\mathsf{WPO}} t_j\theta$. It is obvious if $f > g$. If $f \sim g$ and $[s_1, \dots, s_n]^{\sigma(f)} \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}^{\mathsf{lex}}} [t_1, \dots, t_m]^{\sigma(g)}$, then by the induction hypothesis we get
$$[s_1\theta, \dots, s_n\theta]^{\sigma(f)} \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}^{\mathsf{lex}}} [t_1\theta, \dots, t_m\theta]^{\sigma(g)}$$
Hence, case (2b)–ii applies for $s\theta \mathrel{{}_{(}\!\gtrsim_{)\mathsf{WPO}}} t\theta$. □

Lemma 5.12 (Weak monotonicity) *If $\mathcal{A}$ is weakly monotone and weakly simple with respect to σ, then $\gtrsim_{\mathsf{WPO}}$ is monotone.*

Proof. Suppose $s_i \gtrsim_{\mathsf{WPO}} s_i'$ and let us show
$$s = f(\dots, s_i, \dots) \gtrsim_{\mathsf{WPO}} f(\dots, s_i', \dots) = s'$$
Since $s_i \gtrsim_{\mathcal{A}} s_i'$, we have $s \gtrsim_{\mathcal{A}} s'$ by the weak monotonicity of $\mathcal{A}$.

If $i \notin \sigma(f)$, then we have $[s_1, \dots, s_n]^{\sigma(f)} = [s_1', \dots, s_n']^{\sigma(f)}$. Otherwise, by the weak simplicity assumption we have $s \gtrsim_{\mathcal{A}} s_j$ for every $j \in \sigma(f)$, and thus $s >_{\mathsf{WPO}} s_j$ by case (2a) of Definition 5.2. Hence case (2b)–ii applies for $s \gtrsim_{\mathsf{WPO}} s'$. □

Now we are ready to state the following main result of this chapter.

Theorem 5.13 (Correctness) *If $\mathcal{A}$ is weakly monotone and weakly simple with respect to σ, then $\langle \gtrsim_{\mathsf{WPO}}, >_{\mathsf{WPO}} \rangle$ forms a reduction pair.* □

5.2 Refinements

As we will see in Section 7.5.2, the reduction pair processor induced by Definition 5.2 is already powerful in practice. However in theory, the WPO reduction pair is not a proper extension of the underlying interpretation, e.g., $x \gtrsim_{\mathcal{A}} g(x)$ if $g_{\mathcal{A}}(x) = x$, but $x \gtrsim_{\mathsf{WPO}} g(x)$ cannot hold. Hence we refine the definition of $\gtrsim_{\mathsf{WPO}}$ so as to properly subsume the underlying interpretation.

Note that in the above case, assuming $x \gtrsim_{\mathsf{WPO}} g(x)$ does not cause a problem if g is least with respect to $\gtrsim$ and $\sigma(g) = []$.

Proposition 5.14 *Let $g \in \mathcal{F}$ such that $f \gtrsim g$ for every $f \in \mathcal{F}$ and $\sigma(g) = [\,]$. Then $x \gtrsim_{\mathcal{A}} t = g(t_1, \ldots, t_m)$ implies $s \gtrsim_{\mathsf{WPO}} t[x \mapsto s]$ for arbitrary non-variable terms $s = f(s_1, \ldots, s_n)$.*

Proof. By the definition of $\gtrsim_{\mathcal{A}}$, we have $s \gtrsim_{\mathcal{A}} t[x \mapsto s]$. Since g is least with respect to $\gtrsim$, we have $f \gtrsim g$. Moreover, since $\sigma(g) = [\,]$, we have

$$[s_1, \ldots, s_n]^{\sigma(f)} \gtrsim^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]^{\sigma(g)} = [\,] \qquad \square$$

Proposition 5.14 suggests a refined definition of $s \gtrsim_{\mathsf{WPO}} t$ by adding the following subcase in case 2 of Definition 5.2 (note that $s \gtrsim_{\mathcal{A}} t$ is ensured in this case):

(2c) $s \in \mathcal{V}$ and $t = g(t_1, \ldots, t_m)$ such that $\sigma(g) = [\,]$ and g is least with respect to $\gtrsim$.

Similar refinements are proposed for KBO [45, 72] and for RPO [76], when t is a least constant. Our version is more general since t need not be a constant.

Example 5.15 *[86] Consider a DP problem that induces the following constraints:*

$$\begin{aligned} \mathsf{f}^\sharp(\mathsf{s}(x), y) &> \mathsf{f}^\sharp(\mathsf{p}(\mathsf{s}(x)), \mathsf{p}(y)) \\ \mathsf{f}^\sharp(x, \mathsf{s}(y)) &\gtrsim \mathsf{f}^\sharp(\mathsf{p}(x), \mathsf{p}(\mathsf{s}(y))) \\ \mathsf{p}(\mathsf{s}(x)) &\gtrsim x \end{aligned}$$

Let $\sigma(\mathsf{p}) = [\,]$, $\sigma(\mathsf{f}^\sharp) = [1]$, $\mathsf{s}_{\mathcal{A}}(x) = x + 1$, $\mathsf{p}_{\mathcal{A}}(x) = x$, $\mathsf{f}^\sharp_{\mathcal{A}}(x, y) = x$ and p be least with respect to $\gtrsim$. The first constraint is strictly oriented by case (2b)–i. For the second constraint, it yields $x \gtrsim_{\mathsf{WPO}} \mathsf{p}(x)$, for which case (2c) applies. Note that the argument of p cannot be filtered by an argument filter, because of the third constraint. Hence the refinements of [45] or [76] do not work for this example.

For WPO, a further refinement is also possible when the right-hand side is a variable. Note that if $\sigma(f) = [\,]$, $f(x) \gtrsim_{\mathcal{A}} x$ does not imply $f(x) \gtrsim_{\mathsf{WPO}} x$. Nonetheless, $f(x) \gtrsim_{\mathsf{WPO}} x$ can be assumed if f is *greatest* (i.e., $f \gtrsim g$ for every $g \in \mathcal{F}$), and moreover $\sigma(g) = [\,]$ whenever $f \sim g$. The last condition is crucial, since $f(g(x)) <_{\mathsf{WPO}} g(x)$ if $f \sim g$ and $g = [1]$.

Proposition 5.16 *Suppose that $\mathcal{A}$ is strictly simple with respect to σ, and $f \in \mathcal{F}$ such that either $f > g$, or $f \sim g$ and $\sigma(g) = [\,]$ for every $g \in \mathcal{F}$. Then, $s = f(s_1, \ldots, s_n) \gtrsim_{\mathcal{A}} y$ implies $s[y \mapsto t] \gtrsim_{\mathsf{WPO}} t$ for any non-variable term $t = g(t_1, \ldots, t_m)$.*

Proof. By the definition of $\gtrsim_{\mathcal{A}}$, we have $s[y \mapsto t] \gtrsim_{\mathcal{A}} t$. Moreover by the strict simplicity, $s[y \mapsto t] \gtrsim_{\mathcal{A}} t >_{\mathcal{A}} t_j$ for all $j \in \sigma(g)$. Hence we get $s >_{\mathsf{WPO}} t_j$. If $f > g$, then $s[y \mapsto t] >_{\mathsf{WPO}} t$ by case (2b)–i. If $f \sim g$ and $\sigma(g) = [\,]$, then $s[y \mapsto t] \gtrsim_{\mathsf{WPO}} t$ by case (2b)–ii. □

Provided that $\mathcal{A}$ is strictly simple with respect to σ, Proposition 5.16 suggests a refinement of $s \gtrsim_{\mathsf{WPO}} t$ by adding the following subcase in case 2:

(2d) $s = f(s_1, \ldots, s_n)$ and $t \in \mathcal{V}$ such that for every $g \in \mathcal{F}$, either $f > g$ or $f \gtrsim g$ and $\sigma(g) = [\,]$.

Note that an arbitrary algebra is strictly simple with respect to the empty status. Hence we obtain the following result:

Theorem 5.17 *Consider an instance of WPO that is induced by*

- *a well-founded algebra $\mathcal{A}$ that is* non-trivial*, i.e., there exist $a, b \in A$ such that $a \not\gtrsim_{\mathcal{A}} b$,*
- *the empty status function σ, and*
- *the quasi-precedence $\gtrsim$ such that $f \gtrsim g$ for arbitrary $f, g \in \mathcal{F}$.*

Then, $\langle \gtrsim_{\mathcal{A}}, >_{\mathcal{A}} \rangle = \langle \gtrsim_{\mathsf{WPO}}, >_{\mathsf{WPO}} \rangle$ after the refinements.

Proof. From the definition, it is obvious that $>_{\mathcal{A}} \subseteq >_{\mathsf{WPO}}$ and $\gtrsim_{\mathsf{WPO}} \subseteq \gtrsim_{\mathcal{A}}$. By the assumptions, cases (2b)–i and (2b)–ii of Definition 5.2 cannot apply for $>_{\mathsf{WPO}}$. Hence we easily obtain $>_{\mathsf{WPO}} \subseteq >_{\mathcal{A}}$.

Now suppose $s \gtrsim_{\mathcal{A}} t$ and let us show $s \gtrsim_{\mathsf{WPO}} t$. The proof proceeds by case analysis on the structure of s and t.

- Suppose that both s and t are variables. Then, from non-triviality we have $s = t$, and hence $s \gtrsim_{\mathsf{WPO}} t$.
- Suppose that either s or t is a variable. Then, refinement (2c) or (2d) can be applied to derive $s \gtrsim_{\mathsf{WPO}} t$.
- Suppose $s = f(s_1, \ldots, s_n)$ and $t = f(t_1, \ldots, t_m)$. Since $f > g$ never holds, case (2b)–i of Definition 5.2 can be ignored. Moreover, since $f \gtrsim g$ and $[\,] \gtrsim^{\mathsf{lex}}_{\mathsf{WPO}} [\,]$, $s \gtrsim_{\mathcal{A}} t$ implies $s \gtrsim_{\mathsf{WPO}} t$. □

5.3 Comparison with Other Reduction Pairs

In this section, we compare instances of WPO with existing reduction pairs.

5.3.1 WPO v.s. Argument Filtering

The effect of a partial status has similarity with that of combining argument filtering and a standard total status. Indeed, WPO with partial status subsumes WPO with total status and certain form of argument filtering. An argument filter π is said to be *non-collapsing* iff $\pi(f)$ is a list for every $f \in \mathcal{F}$.

Proposition 5.18 *Let π be a non-collapsing argument filter. For every $\mathcal{F}^\pi$-algebra $\mathcal{A}$ and total status σ on $\mathcal{F}^\pi$, there exists an $\mathcal{F}$-algebra $\mathcal{A}'$ and a partial status σ' on $\mathcal{F}$ such that*

$$\langle \gtrsim^\pi_{\mathsf{WPO}(\mathcal{A},\sigma)}, >^\pi_{\mathsf{WPO}(\mathcal{A},\sigma)} \rangle = \langle \gtrsim_{\mathsf{WPO}(\mathcal{A}',\sigma')}, >_{\mathsf{WPO}(\mathcal{A}',\sigma')} \rangle$$

Proof. Let us define the interpretation of each $f \in \mathcal{F}_n$ in $\mathcal{A}'$ by $f_{\mathcal{A}'}(x_1, \ldots, x_n) := f_{\mathcal{A}}(x_1, \ldots, x_{\pi(f)})$. Then obviously, $\pi(s) \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathcal{A}} \pi(t)$ iff $s \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathcal{A}'} t$. Moreover, we define $\sigma'(f)$ by $\pi(f) \star \sigma(f)$, where $\star$ is a left-associative operator defined by

$$[a_1, \ldots, a_n] \star [i_1, \ldots, i_{n'}] := [a_{i_1}, \ldots, a_{i_{n'}}]$$

Now we verify that $s \mathrel{{}_{(}{\gtrsim}_{)}}^\pi_{\mathsf{WPO}(\mathcal{A},\sigma)} t$ implies $s \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$ by induction on $|s| + |t|$. If $s \in \mathcal{V}$, then $s = \pi(t) = t$ since π is non-collapsing, and hence $s \gtrsim_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$. Suppose $s = f(s_1, \ldots, s_n)$. We proceed by case splitting for the derivation of $\pi(s) \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathsf{WPO}(\mathcal{A},\sigma)} \pi(t)$.

1. Suppose $\pi(s) >_{\mathcal{A}} \pi(t)$. We obviously have $s >_{\mathcal{A}'} t$ and hence $s >_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$.

2. Suppose $\pi(s) \gtrsim_{\mathcal{A}} \pi(t)$. We obviously have $s \gtrsim_{\mathcal{A}'} t$.

   (a) Suppose that $\pi(s) \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathsf{WPO}(\mathcal{A},\sigma)} \pi(t)$ is derived by case (2a). Then we have $s_i \gtrsim^\pi_{\mathsf{WPO}(\mathcal{A},\sigma)} t$ for some $i \in [1, \ldots, n] \star \pi(f)$. By the induction hypothesis we have $s_i \gtrsim_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$, and since $i \in \sigma'(f)$, $s >_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$ by case (2a).

   (b) Suppose that $\pi(s) \mathrel{{}_{(}{\gtrsim}_{)}}_{\mathsf{WPO}(\mathcal{A},\sigma)} \pi(t)$ is derived by case (2b). Then we have $t = g(t_1, \ldots, t_m)$ and $s >^\pi_{\mathsf{WPO}(\mathcal{A},\sigma)} t_j$ for every $j \in [1, \ldots, m] \star \pi(g)$. By the induction hypothesis we have $s >_{\mathsf{WPO}(\mathcal{A}',\sigma')} t_j$, for all $j \in \sigma'(f)$.

If furthermore $f > g$, then immediately $s >_{\mathsf{WPO}(\mathcal{A}',\sigma')} t$ by case (2b)–i. If case (2b)–ii applies, then we obtain

$$[s_1, \dots, s_n] \star \pi(f) \star \sigma(f) \mathrel{\substack{>\\(\sim)}}{}^{\pi\,\mathsf{lex}}_{\mathsf{WPO}(\mathcal{A},\sigma)} [t_1, \dots, t_m] \star \pi(g) \star \sigma(g)$$

By the induction hypothesis and definition of σ' we obtain

$$[s_1, \dots, s_n]^{\sigma'(f)} \mathrel{\substack{>\\(\sim)}}{}_{\mathsf{WPO}(\mathcal{A}',\sigma')} [t_1, \dots, t_m]^{\sigma'(g)} \qquad \square$$

In Definition 5.2, it is obvious that induced $>_{\mathsf{WPO}}$ is identical to that induced by Definition 4.1, if we choose a *total* status σ. Hence Theorems 4.6 and 4.18 imply that WPO subsumes GKBO and LPO also as a reduction pair.

5.3.2 WPO v.s. Interpretation Methods

Note that Proposition 4.12 does not imply that WPO($\mathcal{P}ol$) subsumes POLO as a reduction pair, since the "weak-part" $\geq_{\mathcal{P}ol}$ is not considered in the theorem. Nonetheless, after the refinements in Section 5.2, we obtain the following result from Theorem 5.17:

Corollary 5.19 *The reduction pair WPO($\mathcal{P}ol$) with the refinements (2c) and (2d) subsumes POLO.* $\square$

It is now easy to obtain the following result:

Corollary 5.20 *The reduction pair WPO($\mathcal{MP}ol$) with the refinements (2c) and (2d) subsumes POLO, KBO, TKBO and LPO.* $\square$

Moreover, WPO also subsumes the *matrix interpretation method* [20], when weights are computed by a matrix interpretation $\mathcal{M}at$. Note that a matrix interpretation is not always weakly simple. Hence as a *reduction order*, Definition 4.1 cannot be applied for $\mathcal{M}at$ in general. The situation is relaxed for reduction pairs, and from Theorem 5.17 we obtain the following.

Corollary 5.21 *The reduction pair induced by WPO($\mathcal{M}at$) subsumes the reduction pair induced by the matrix interpretation $\mathcal{M}at$.* $\square$

5.3.3 WPO v.s. RPOLO

Finally, we show that WPO($\mathcal{MP}ol$) and *RPOLO* of Bofill *et al.* [8], another approach to unifying LPO and POLO, are incomparable in theory.

In order to define RPOLO, we need some preliminaries. The signature $\mathcal{F}$ is split into two disjoint sets: the set $\mathcal{F}_{\mathsf{POLO}}$ of *POLO-symbols* and the set $\mathcal{F}_{\mathsf{RPO}}$ of *RPO-symbols*. The polynomial interpretation I of RPOLO introduces extra fresh variables to represent terms rooted by an RPO-symbol.

$$I(s) = \begin{cases} f_I(I(s_1), \ldots, I(s_n)) & \text{if } s = f(s_1, \ldots, s_n) \text{ and } f \in \mathcal{F}_{\mathsf{POLO}} \\ v_s & \text{otherwise} \end{cases}$$

where each $f_I(x_1, \ldots, x_n)$ is a polynomial expression over variables $x_1, \ldots, x_n$, and v_s is the fresh variable representing s.

The set of *accessible terms* are defined as follows:

$$\mathsf{Acc}(s) = \begin{cases} \mathsf{Acc}(s_1) \cup \cdots \cup \mathsf{Acc}(s_n) & \text{if } f(s_1, \ldots, s_n) \text{ and } f \in \mathcal{F}_{\mathsf{POLO}} \\ \{s\} & \text{otherwise} \end{cases}$$

Let ϕ be a formula built from atoms of the form $p > q$ or $p \geq q$, where p and q are polynomial expressions over variables from X. We write $\models \phi$ iff every assignment $\alpha : X \to \mathbb{N}$ satisfies ϕ.

Definition 5.22 (RPOLO [8]**)** *The pair $\langle \gtrsim_{\mathsf{RPOLO}}, \gtrsim_{\mathsf{RPOLO}} \rangle$ is defined as follows: $x \gtrsim_{\mathsf{RPOLO}} x$ and $s = f(s_1, \ldots, s_n) \ {}_{(\sim)}^{>}{}_{\mathsf{RPOLO}}\ t$ iff*

1. *$f \in \mathcal{F}_{\mathsf{POLO}}$ and $I(s) >_{C(s)} I(t)$,*
2. *$f \in \mathcal{F}_{\mathsf{RPO}}$ and*
   - *(a) $\exists i \in \{1, \ldots, n\}.\ s_i \gtrsim_{\mathsf{RPOLO}} t$, or*
   - *(b) $t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{\mathsf{POLO}}$ and $\forall t' \in \mathsf{Acc}(t).\ s >_{\mathsf{RPOLO}} t'$, or*
   - *(c) $t = g(t_1, \ldots, t_m)$, $g \in \mathcal{F}_{\mathsf{RPO}}$, $\forall j \in \{1, \ldots, m\}.\ s >_{\mathsf{RPOLO}} t_j$, and*
     - *i. $f > g$, or*
     - *ii. $f \gtrsim g$ and $[s_1, \ldots, s_n] \ {}_{(\sim)}^{>}{}^{\mathsf{lex}}_{\mathsf{RPOLO}}\ [t_1, \ldots, t_m]$,*

where $s >_{C(u)} t$ holds iff $\models \phi \Rightarrow s > t$ for some conjunction ϕ of atoms of the following forms:

- *$v_s > t$ where $s \in \mathsf{Acc}(u)$, t is a polynomial expression over variables $v_{t_1}, \ldots, v_{t_m}$ such that $s >_{\mathsf{RPOLO}} t_j$ for every $j \in \{1, \ldots, m\}$, or*

- *$v_s \geq v_t$ where $s \in \mathsf{Acc}(u)$, $\mathsf{root}(v_t) \in \mathcal{F}_{\mathsf{RPO}}$ and $s \gtrsim_{\mathsf{RPOLO}} t$.*

Note that in the definition of RPOLO, the relations $\gtrsim_{\mathsf{RPOLO}}$, $>_{\mathsf{RPOLO}}$, and $>_{C(u)}$ are defined by mutual recursion.

Theorem 5.23 ([8]) *The pair $\langle \gtrsim_{\mathsf{RPOLO}}, >_{\mathsf{RPOLO}} \rangle$ forms a reduction pair.* □

Now we verify that WPO($\mathcal{MP}ol$) is not subsumed by RPOLO. More precisely, we show that RPOLO does not subsume even KBO.

Example 5.24 *We show that the constraint $\mathsf{f}(\mathsf{g}(x)) \succ \mathsf{g}(\mathsf{f}(\mathsf{f}(x)))$ cannot be satisfied by RPOLO.*[2] *Note that this constraint is satisfied by KBO with $w(\mathsf{f}) = 0$ and $\mathsf{f} > \mathsf{g}$.*

- *Suppose $f \in \mathcal{F}_{\mathsf{POLO}}$. Since this constraint cannot be satisfied by POLO, g must be in $\mathcal{F}_{\mathsf{RPO}}$. Hence we need $\mathsf{f}_{\mathcal{P}ol}(v_{\mathsf{g}(x)}) >_{C(\mathsf{f}(\mathsf{g}(x)))} v_{\mathsf{g}(\mathsf{f}(\mathsf{f}(x)))}$. This requires either*
  - *$\mathsf{f}_{\mathcal{P}ol}(x) = x$ and $\mathsf{g}(x) >_{\mathsf{RPOLO}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x)))$, or*
  - *$\mathsf{f}_{\mathcal{P}ol}(x) > x$ and $\mathsf{g}(x) \geq_{\mathsf{RPOLO}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x)))$.*

  In either case, we obtain $\mathsf{g}(x) >_{\mathsf{RPOLO}} \mathsf{g}(x)$, which is a contradiction.

- *Suppose $\mathsf{f} \in \mathcal{F}_{\mathsf{RPO}}$. Since this constraint cannot be satisfied by RPO, g must be in $\mathcal{F}_{\mathsf{POLO}}$. Hence we need*
  - *$\mathsf{g}(x) \geq_{\mathsf{RPOLO}} \mathsf{g}(\mathsf{f}(\mathsf{f}(x)))$, or*
  - *$\mathsf{f}(\mathsf{g}(x)) >_{\mathsf{RPOLO}} \mathsf{f}(\mathsf{f}(x))$.*

  The first case contradicts $\mathsf{f}(x) >_{\mathsf{RPOLO}} x$. The second case contradicts the fact that $\mathsf{f}(x) >_{\mathsf{RPOLO}} \mathsf{g}(x)$.

On the other hand, RPOLO is also not subsumed by WPO($\mathcal{MP}ol$), as the following example illustrates.

[2] To simplify the discussion, we do not consider possibility for *argument filterings* or *usable rules* [2] in the following examples. Nonetheless, it is easy to exclude these techniques by adding rules e.g. $\mathsf{g}(x) \to x$.

Example 5.25 *Consider a DP problem that induces the following constraints:*

$$\mathsf{f}^\sharp(\mathsf{i}(x, \mathsf{i}(y, \mathsf{g}(z)))) \succ \mathsf{f}^\sharp(\mathsf{i}(y, \mathsf{i}(z, x))) \tag{5.1}$$

$$\mathsf{f}(\mathsf{g}(\mathsf{h}(x))) \succsim \mathsf{g}(\mathsf{f}(\mathsf{h}(\mathsf{g}(x)))) \tag{5.2}$$

$$\mathsf{i}(y, \mathsf{i}(z, x)) \succsim \mathsf{i}(x, \mathsf{i}(y, z)) \tag{5.3}$$

where constraint (5.2) *is from [92, Proposition 10].*

- *First, let us show that the set of constraints cannot be satisfied by WPO($\mathcal{MP}ol$). Since* f*,* g *and* h *are unary, we only consider* $ws(\mathsf{f}) = ws(\mathsf{g}) = ws(\mathsf{h}) = \mathsf{pol}$. *It is easy to adjust [92, Proposition 10] to show that* $\mathsf{g}_{\mathcal{P}ol}(x) = x$ *whenever* (5.2) *is satisfied. Together with* (5.3)*, we obtain*

  $$\mathsf{i}(y, \mathsf{i}(z, x)) \geq_{\mathcal{MP}ol} \mathsf{i}(x, \mathsf{i}(y, z)) =_{\mathcal{MP}ol} \mathsf{i}(x, \mathsf{i}(y, \mathsf{g}(z)))$$

  Hence, case 1 of Definition 5.2 cannot be applied for constraint (5.1)*. Moreover, by any choice of* $\sigma(\mathsf{i})$*, case (2b)–ii cannot be applied, either.*

- *Second, we show that the set of constraints can be satisfied by RPOLO. Consider* $\mathsf{f}, \mathsf{g}, \mathsf{h} \in \mathcal{F}_{\mathsf{RPO}}$*,* $\mathsf{i}, \mathsf{f}^\sharp \in \mathcal{F}_{\mathsf{POLO}}$*,* $\mathsf{f} > \mathsf{g} > \mathsf{h}$*,* $\mathsf{i}_{\mathcal{P}ol}(x, y) = x + y$ *and* $\mathsf{f}^\sharp_{\mathcal{P}ol}(x) = x$*. Then, constraint* (5.2) *is strictly oriented and* (5.3) *is weakly oriented. Since* $\mathsf{g}(z) >_{\mathsf{RPOLO}} z$ *and* $v_{\mathsf{g}(z)} > z$ *implies* $x + y + v_{\mathsf{g}(z)} > y + z + x$*, constraint* (5.1) *is also satisfied.* □

Chapter 6

SMT Encoding

In the preceding chapters, we have concentrated on theoretical aspects. In this chapter, we consider how to implement the instances of WPO using SMT solvers. In particular, WPO($\mathcal{S}um$), WPO($\mathcal{M}ax$), and WPO($\mathcal{MS}um$) are reduced to SMT problems over linear arithmetic, and as a consequence, decidability is ensured for orientability problems of these orders. To this end, we extend the corresponding approach for KBO proposed by Zankl et al. [91].

Definition 6.1 *An* expression *e is built from (non-negative integer) variables, constants and the binary symbols $\cdot$ and $+$ denoting multiplication and addition, respectively. A* formula *is built from atoms of the form $e_1 > e_2$ and $e_1 \geq e_2$, negation $\neg$, and the binary symbols $\wedge$, $\vee$, and $\Rightarrow$ denoting conjunction, disjunction and implication, respectively. The precedence of these symbols is in the order we listed above.*

The main interest of the SMT encoding approach is to employ SMT solvers for finding a concrete algebra that proves termination of a given TRS or finiteness of a given DP problem. Hence we assume that algebras are *parameterized* by a set of expression variables.

Definition 6.2 *An algebra $\mathcal{A}$* parameterized *by a set V of variables is a mapping that induces a concrete algebra $\mathcal{A}^\alpha$ from an assignment α whose domain contains V. An* encoding *of the relation $\underset{(\sim)}{>}_{\mathcal{A}}$ is a function that assigns for two terms s and t a formula $[\![s \underset{(\sim)}{>}_{\mathcal{A}} t]\!]$ over variables from V such that $\alpha \models [\![s \underset{(\sim)}{>}_{\mathcal{A}} t]\!]$ iff $s \underset{(\sim)}{>}_{\mathcal{A}^\alpha} t$.*

In the encodings presented in the following sections, we consider $\mathcal{A}$ to be parameterized by at most the following variables:

- integer variables w_f and w_0 denoting w_f and w_0, respectively, and
- integer variables $\mathsf{sc}_{f,i}$ and $\mathsf{sp}_{f,i}$ denoting $c_{f,i}$ and $p_{f,i}$, respectively.

6.1 The Common Structure

To optimize the presentation, we first present an encoding of the common structure of WPO independent from the shape of $\mathcal{A}$. Hence, we assume encodings for $>_{\mathcal{A}}$ and $\gtrsim_{\mathcal{A}}$ are given.

Following [91], we first represent a precedence $\gtrsim$ by integer variables p_f.

Definition 6.3 *Let α be an assignment such that $\alpha(\mathsf{p}_f) \in \mathbb{N}$ is defined for every $f \in \mathcal{F}$. The* precedence $\gtrsim^\alpha$ induced by α *is defined as follows: $f \gtrsim^\alpha g$ iff $\alpha \models \mathsf{p}_f \geq \mathsf{p}_g$.*

Next, we consider representing a partial status by imitating the encoding of a filtered permutation proposed in [12]. We introduce the boolean variables $\mathsf{st}_{f,i}$ and $\mathsf{st}_{f,i,j}$, so that an assignment α induces a status σ^α as follows: $\alpha \models \mathsf{st}_{f,i}$ iff $i \in \sigma^\alpha(f)$; and $\alpha \models \mathsf{st}_{f,i,j}$ iff i is the j-th element in $\sigma^\alpha(f)$. In order for σ^α to be well-defined, every $i \in \sigma^\alpha(f)$ must occur exactly once in $\sigma^\alpha(f)$ and any $i \notin \sigma^\alpha(f)$ must not occur in $\sigma^\alpha(f)$. These conditions are represented by the following formula:

$$\mathsf{ST} := \bigwedge_{f\in\mathcal{F}_n} \bigwedge_{i=1}^{n} \Big(\Big(\mathsf{st}_{f,i} \Rightarrow \sum_{j=1}^{n} \mathsf{st}_{f,i,j} = 1 \Big) \land \Big(\neg \mathsf{st}_{f,i} \Rightarrow \sum_{j=1}^{n} \mathsf{st}_{f,i,j} = 0 \Big) \Big)$$

It is easy to verify that σ^α is well-defined if $\alpha \models \mathsf{ST}$. In contrast to the previous works [91, 12], we moreover introduce the following formula to ensure weak simplicity of $\mathcal{A}$ with respect to σ:

$$\mathsf{SIMP} := \bigwedge_{f\in\mathcal{F}_n} \bigwedge_{i=1}^{n} \Big(\mathsf{st}_{f,i} \Rightarrow [\![f(x_1,\ldots,x_n) \gtrsim_{\mathcal{A}} x_i]\!] \Big)$$

Note that in the formula SIMP, the condition $f(x_1,\ldots,x_n) \gtrsim_{\mathcal{A}} x_i$ can often be encoded in a more efficient way. For linear polynomial interpretations, for example, this condition is equivalent to $\mathsf{sc}_{f,i} \geq 1$.

Lemma 6.4 *Let $\mathcal{A}$ be a well-founded algebra parameterized by V and α an assignment on $V \cup \{\mathsf{st}_{f,i}, \mathsf{st}_{f,i,j} \mid f \in \mathcal{F}_n,\ i,j \in \{1,\ldots,n\}\}$. Then $\alpha \models \mathsf{ST} \land \mathsf{SIMP}$ iff $\mathcal{A}^\alpha$ is weakly simple with respect to the partial status σ^α.* □

Now we present the encodings for WPO.

Definition 6.5 *The encodings of $>_{\mathsf{WPO}}$ and $\gtrsim_{\mathsf{WPO}}$ are defined as follows:*

$$[\![s \mathrel{_{(}\!\gtrsim\!_{)}}_{\mathsf{WPO}} t]\!] := [\![s >_{\mathcal{A}} t]\!] \vee \big([\![s \gtrsim_{\mathcal{A}} t]\!] \wedge s \mathrel{_{(}\!\gtrsim\!_{)}}_1 t \big)$$

where the formula $s \mathrel{_{(}\!\gtrsim\!_{)}}_1 t$ is defined as follows:

$$\begin{array}{rcl}
x \gtrsim_1 t & := & \begin{cases} \textsc{True} & \textit{if } x = t \\ \textsc{False} & \textit{otherwise} \end{cases} \\
x >_1 t & := & \textsc{False} \\
f(s_1, \ldots, s_n) \mathrel{_{(}\!\gtrsim\!_{)}}_1 t & := & \displaystyle\bigvee_{i=1}^{n} \Big(\mathsf{st}_{f,i} \wedge [\![s_i \gtrsim_{\mathsf{WPO}} t]\!] \Big) \vee f(s_1, \ldots, s_n) \mathrel{_{(}\!\gtrsim\!_{)}}_2 t
\end{array}$$

and the formula $f(s_1, \ldots, s_n) \mathrel{_{(}\!\gtrsim\!_{)}}_2 t$ is defined as follows:

$$\begin{array}{l}
f(s_1, \ldots, s_n) \mathrel{_{(}\!\gtrsim\!_{)}}_2 y := \textsc{False} \\
f(s_1, \ldots, s_n) \mathrel{_{(}\!\gtrsim\!_{)}}_2 g(t_1, \ldots, t_m) := \displaystyle\bigwedge_{j=1}^{m} \Big(\mathsf{st}_{g,j} \Rightarrow [\![s >_{\mathsf{WPO}} t_j]\!] \Big) \wedge \\
\qquad \Big(\mathsf{p}_f > \mathsf{p}_g \vee \mathsf{p}_f = \mathsf{p}_g \wedge [\![[s_1, \ldots, s_n]^{\sigma(f)} \mathrel{_{(}\!\gtrsim\!_{)}}^{\mathsf{lex}}_{\mathsf{WPO}} [t_1, \ldots, t_m]^{\sigma(g)}]\!] \Big)
\end{array}$$

Here we do not present the encoding for the lexicographic extension with respect to permutation, which can be found in [12]. In the definition, $s >_1 t$ indicates that $s >_{\mathsf{WPO}} t$ is derived by cases (2a) or (2b), and $s >_2 t$ indicates that $s >_{\mathsf{WPO}} t$ is derived by cases (2b)–i or (2b)–ii.

For an assignment α, we write $\mathrel{_{(}\!\gtrsim\!_{)}}^{\alpha}_{\mathsf{WPO}}$ to denote the instance of WPO corresponding to α, i.e., $\mathrel{_{(}\!\gtrsim\!_{)}}^{\alpha}_{\mathsf{WPO}}$ is induced by the algebra $\mathcal{A}^{\alpha}$, the precedence $\gtrsim^{\alpha}$ and the partial status σ^{α}.

Lemma 6.6 *Let $\alpha \models \mathsf{ST} \wedge \mathsf{SIMP}$. Then $\alpha \models [\![s \mathrel{_{(}\!\gtrsim\!_{)}}_{\mathsf{WPO}} t]\!]$ iff $s \mathrel{_{(}\!\gtrsim\!_{)}}^{\alpha}_{\mathsf{WPO}} t$.*

Proof. Obvious. □

Theorem 6.7 *If the following formula is satisfiable:*

$$\mathsf{ST} \wedge \mathsf{SIMP} \wedge \bigwedge_{l \to r \in \mathcal{R} \cup \mathcal{P}} [\![l \gtrsim_{\mathsf{WPO}} r]\!] \wedge \bigvee_{l \to r \in \mathcal{P}'} [\![l >_{\mathsf{WPO}} r]\!] \qquad (6.1)$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{ \langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle \}$ is sound.

Proof. Let α be the assignment that satisfies (6.1). By Lemma 6.6, we obtain $l \gtrsim^{\alpha}_{\mathsf{WPO}} r$ for all $l \to r \in \mathcal{R} \cup \mathcal{P}$ and $l >^{\alpha}_{\mathsf{WPO}} r$ for all $l \to r \in \mathcal{P}'$. Moreover by Theorem 5.13, $\langle \gtrsim^{\alpha}_{\mathsf{WPO}}, >^{\alpha}_{\mathsf{WPO}} \rangle$ forms a reduction pair. Hence Theorem 2.45 concludes the soundness of this DP processor. □

In the following sections, we give encodings depending on the choice of $\mathcal{A}$ for each instance of WPO.

6.2 Encoding WPO($\mathcal{P}ol$) and WPO($\mathcal{S}um$)

First we present encodings for linear polynomial interpretation $\mathcal{P}ol$. The encodings for $\mathcal{S}um$ is obtained by fixing coefficients to 1. The weight of a term s and the variable coefficient of x in s are encoded as follows:

$$\mathsf{w}(s) := \begin{cases} \mathsf{w}_0 & \text{if } s \in \mathcal{V} \\ \mathsf{w}_f + \displaystyle\sum_{i=1}^{n} \mathsf{sc}_{f,i} \cdot \mathsf{w}(s_i) & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

$$\mathsf{vc}(x, s) := \begin{cases} 1 & \text{if } x = s \\ 0 & \text{if } x \neq s \in \mathcal{V} \\ \displaystyle\sum_{i=1}^{n} \mathsf{sc}_{f,i} \cdot \mathsf{vc}(x, s_i) & \text{if } s = f(s_1, \ldots, s_n) \end{cases}$$

We have to ensure w_0 to be the lower bound of weights of terms. To ensure $\mathsf{w}(f(s_n)) > \mathsf{w}_0$ for every term $f(s_n)$, we need either $\mathsf{w}_f \geq \mathsf{w}_0$ or one of the arguments to have a positive coefficient (note that the weight of this argument is at least w_0). This is represented by

$$\mathsf{WMIN} := \bigwedge_{f \in \mathcal{F}_n} \Big(\mathsf{w}_f \geq \mathsf{w}_0 \vee \bigvee_{i=1}^{n} \mathsf{sc}_{f,i} \geq 1 \Big)$$

Now the relations $>_{\mathcal{P}ol}$ and $\geq_{\mathcal{P}ol}$ are encoded as follows:

$$[\![s \mathrel{{}_{(}\geq_{)}}_{\mathcal{P}ol} t]\!] := \mathsf{w}(s) \mathrel{{}_{(}\geq_{)}} \mathsf{w}(t) \wedge \bigwedge_{x \in \mathsf{Var}(t)} \mathsf{vc}(x, s) \geq \mathsf{vc}(x, t)$$

Corollary 6.8 *If the following formula is satisfiable:*

$$\mathsf{ST} \wedge \mathsf{SIMP} \wedge \mathsf{WMIN} \wedge \bigwedge_{l \to r \in \mathcal{R} \cup \mathcal{P}} [\![l \gtrsim_{\mathsf{WPO}(\mathcal{P}ol)} r]\!] \wedge \bigvee_{l \to r \in \mathcal{P}'} [\![l >_{\mathsf{WPO}(\mathcal{P}ol)} r]\!]$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound. □

6.3 Encoding WPO($\mathcal{M}ax$)

In this section, we consider encoding WPO($\mathcal{M}ax$). Unfortunately, we are aware of no SMT solver which supports a built-in max operator. Hence we consider encoding the constraint $s >_{\mathcal{M}ax} t$ into both quantified and quantifier-free formulas.

First, we present an encoding to a quantified formula. A straightforward encoding would involve

$$\mathsf{w}(s) := \begin{cases} s & \text{if } s \in \mathcal{V} \\ v & \text{if } s = f(s_1, \dots, s_n) \end{cases}$$

where v is a fresh integer variable representing $\max\{\mathsf{w}_f, \mathsf{w}(s_1), \dots, \mathsf{w}(s_n)\}$ with the following constraint ϕ added into the context:

$$\phi := v \geq \mathsf{w}_f \wedge \bigwedge_{i=1}^{n} v \geq \mathsf{w}(s_i) \wedge \Big(v = \mathsf{w}_f \vee \bigvee_{i=1}^{n} v = \mathsf{w}(s_i)\Big)$$

Then the constraint $s \mathrel{{}_{(}\!\geq_{)}}_{\mathcal{M}ax} t$ can be encoded as follows:

$$[\![s \mathrel{{}_{(}\!\geq_{)}}_{\mathcal{M}ax} t]\!] := \forall x_1, \dots, x_k, v_1, \dots, v_m.\ \phi_1 \wedge \cdots \wedge \phi_m \Rightarrow \mathsf{w}(s) \mathrel{{}_{(}\!\geq_{)}} \mathsf{w}(t)$$

where $\{x_1, \dots, x_k\} = \mathsf{Var}(s) \cup \mathsf{Var}(t)$ and each $\langle \phi_j, v_j \rangle$ is the pair of the constraint and the fresh variable introduced during the encoding.

Although quantified linear integer arithmetic (i.e., Presburger arithmetic) is well-known to be decidable, the SMT solvers we have tested could not solve the problems generated by the above straightforward encoding efficiently, if at all. Fuhs *et al.* [24] proposed a sound elimination of quantifiers by introducing new template polynomials. Here we propose another encoding to quantifier-free formulas that does not introduce extra polynomials and is sound and complete for linear polynomials with max.

Definition 6.9 *A* generalized weight *[46] is a pair $\langle n, N \rangle$ where $n \in \mathbb{N}$ and N is a finite multiset*[1] *over $\mathcal{V}$. We define the following operations:*

$$\langle n, N \rangle + \langle m, M \rangle := \langle n + m, N \uplus M \rangle$$
$$n \cdot \langle m, M \rangle := \langle n \cdot m, n \cdot M \rangle$$

where $n \cdot M$ denotes the multiset that maps x to $n \cdot M(x)$ for every $x \in \mathcal{V}$. We encode a generalized weight as a pair of an expression and a mapping N from $\mathcal{V}$ to expressions such that the domain $\mathsf{Dom}(N) := \{x \mid N(x) \neq 0\}$ *of N is finite.*

[1]In the encoding for $\mathcal{M}ax$, N need not contain more than one variable. This generality is reserved for the encoding of $\mathcal{MP}ol$.

A generalized weight $\langle n, N\rangle$ represents the expression $n+\sum_{x\in N} x$. Notations for generalized weights are naturally extended for encoded ones. The relation $\supseteq$ on multisets is encoded as follows:

$$N \supseteq M := \bigwedge_{x\in\mathsf{Dom}(M)} N(x) \geq M(x)$$

Now we consider removing max.

Definition 6.10 *The* expanded weight $\overline{\mathsf{w}}(s)$ *of a term s is a set of generalized weights, which is defined as follows:*

$$\overline{\mathsf{w}}(s) := \begin{cases} \{\langle \mathsf{w}_0, \{s\}\rangle\} & \text{if } s \in \mathcal{V} \\ \{\langle \mathsf{w}_f, \emptyset\rangle\} \cup \{\mathsf{sp}_{f,i} + p \mid p \in \overline{\mathsf{w}}(s_i),\ 1 \leq i \leq n\} & \text{if } s = f(s_1, \dots, s_n) \end{cases}$$

The expanded weight $\overline{\mathsf{w}}(s) = \{p_1, \dots, p_n\}$ represents the expression $\max_{i=1}^{n} e_i$, where each generalized weight p_i represents the expression e_i. Using expanded weights, we can encode $>_{\mathcal{M}ax}$ and $\geq_{\mathcal{M}ax}$ in a way similar to the *max set ordering* presented in [6]:

$$[\![s \ _{(}\geq_{)\mathcal{M}ax}\ t]\!] := \bigwedge_{\langle m,M\rangle\in\overline{\mathsf{w}}(t)} \bigvee_{\langle n,N\rangle\in\overline{\mathsf{w}}(s)} (n \ _{(}\geq_{)}\ m \wedge N \supseteq M)$$

Using the quantified or quantifier-free encodings, we obtain the following corollary of Theorem 5.13:

Corollary 6.11 *If the following formula is satisfiable:*

$$\mathsf{ST} \wedge \bigwedge_{l\to r\in\mathcal{R}\cup\mathcal{P}} [\![l \gtrsim_{\mathsf{WPO}(\mathcal{M}ax)} r]\!] \ \wedge \bigvee_{l\to r\in\mathcal{P}'} [\![l >_{\mathsf{WPO}(\mathcal{M}ax)} r]\!]$$

then the DP processor that maps $\langle\mathcal{P}, \mathcal{R}\rangle$ to $\{\langle\mathcal{P}\setminus\mathcal{P}', \mathcal{R}\rangle\}$ is sound. □

6.4 Encoding WPO($\mathcal{MP}ol$) and WPO($\mathcal{MS}um$)

In this section, we consider encoding linear polynomials with max into SMT formulas. First, we extend Definition 6.10 to admit a weight status.

Definition 6.12 *For a weight status ws, the* expanded weight $\overline{\mathsf{w}}^{ws}(s)$ *of a term s is the set of generalized weight, which is recursively defined as follows:*

$$\overline{\mathsf{w}}^{ws}(s) := \begin{cases} \{(\mathsf{w}_0, \{s\})\} & \text{if } s \in \mathcal{V} \\ S & \text{if } s = f(s_1, \dots, s_n),\ ws(f) = \mathsf{pol} \\ T & \text{if } s = f(s_1, \dots, s_n),\ ws(f) = \mathsf{max} \end{cases}$$

where

$$S = \left\{ \mathsf{w}_f + \sum_{i=1}^{n} \mathsf{sc}_{f,i} \cdot p_i \;\middle|\; p_1 \in \overline{\mathsf{w}}^{ws}(s_1), \ldots, p_n \in \overline{\mathsf{w}}^{ws}(s_n) \right\}$$
$$T = \{\mathsf{w}_f\} \cup \{\mathsf{sp}_{f,i} + \mathsf{sc}_{f,i} \cdot p \mid p \in \overline{\mathsf{w}}^{ws}(s_i),\ i \in \{1, \ldots, n\}\}$$

Now the encoding of $>_{\mathcal{MP}ol}$ and $\geq_{\mathcal{MP}ol}$ are given as follows:

$$[\![s \;{}_{(}\geq_{)}{}_{\mathcal{MP}ol}\; t]\!] := \bigwedge_{\langle m,M\rangle \in \overline{\mathsf{w}}^{ws}(t)} \; \bigvee_{\langle n,N\rangle \in \overline{\mathsf{w}}^{ws}(s)} \big(n \;{}_{(}\geq_{)}\; m \wedge N \supseteq M \big)$$

Corollary 6.13 *If the following formula is satisfiable:*

$$\mathsf{ST} \wedge \mathsf{SIMP} \wedge \mathsf{WMIN} \wedge \bigwedge_{l \to r \in \mathcal{R} \cup \mathcal{P}} [\![l \gtrsim_{\mathsf{WPO}(\mathcal{MP}ol)} r]\!] \;\wedge\; \bigvee_{l \to r \in \mathcal{P}'} [\![l >_{\mathsf{WPO}(\mathcal{MP}ol)} r]\!]$$

then the DP processor that maps $\langle \mathcal{P}, \mathcal{R} \rangle$ to $\{\langle \mathcal{P} \setminus \mathcal{P}', \mathcal{R} \rangle\}$ is sound. □

6.5 Encoding WPO($\mathcal{M}at$)

We omit presenting an encoding of the matrix interpretation method, which can be found in [20]. In order to use a matrix interpretation in WPO, however, small care is needed; one has to ensure weak simplicity of $\mathcal{M}at$ with respect to σ. This can be done by ensuring that $C_{f,i} \geq E$ for every $f \in \mathcal{F}_n$ and $i \in \sigma(f)$, where E denotes the unit matrix.

Lemma 6.14 *If $C_{f,i}^{j,j} \geq 1$ for all $f \in \mathcal{F}_n$, $i \in \sigma(f)$ and $j \in \{1, \ldots, d\}$, then $\mathcal{M}at$ is weakly simple with respect to σ.* □

6.6 Encoding for Reduction Orders

In case one wants an encoding for the reduction order $>_{\mathsf{WPO}}$ defined in Definition 4.1, the status σ must be total. This can be ensured by enforcing $i \in \sigma(f)$ for all $i \in \{1, \ldots, n\}$ and $f \in \mathcal{F}$, which is represented by the following formula:

$$\mathsf{TOTAL} := \bigwedge_{f \in \mathcal{F}_n} \bigwedge_{i=1}^{n} \mathsf{st}_{f,i}$$

or equivalently by replacing all $\mathsf{st}_{f,i}$ by True. Note that $\mathsf{TOTAL} \wedge \mathsf{SIMP}$ enforces all the subterm coefficients to be greater than 1.

Theorem 6.15 *If the following formula is satisfiable:*

$$\mathsf{TOTAL} \wedge \mathsf{ST} \wedge \mathsf{SIMP} \wedge \mathsf{WMIN} \wedge \bigwedge_{l \to r \in \mathcal{R}} [\![l >_{\mathsf{WPO}(\mathcal{MPol})} r]\!]$$

then $\mathcal{R}$ is orientable by WPO($\mathcal{MPol}$). □

6.7 Optimizations

In our implementation, some optimizations are performed during the encoding. For example, formulas like FALSE $\wedge\ \phi$ are reduced in advance to avoid generating meaningless formulas, and temporary variables are inserted to avoid multiple occurrences of an expression or a formula. Moreover, we apply several optimizations that we discuss below.

6.7.1 Fixing w_0

We can simplify the encoded formulas by fixing w_0. For KBO, Winkler *et al.* [81] show that w_0 can be fixed to arbitrary $k > 0$ e.g. 1 without loosing the power of the order. Applying their technique, it can be shown that for WPO($\mathcal{S}um$), w_0 can be fixed to 0. On the contrary to KBO, however, w_0 cannot be fixed to $k > 0$ since transforming a weight function $\langle w, 0\rangle$ into $\langle w^k, k\rangle$ may assign negative weights to some symbols.

6.7.2 Fixing Weight Status

For POLO and WPO using algebras $\mathcal{MS}um$ and $\mathcal{MP}ol$, it may be not practical to consider all possible weight statuses. Hence, we introduce a heuristic for fixing ws. In case of WPO($\mathcal{MS}um$), ws should at least satisfy the following condition for all $l \to r \in \mathcal{R} \cup \mathcal{P}$:

$$\bigwedge_{\langle m,M\rangle \in \overline{\mathsf{w}}^{ws}(r)} \bigvee_{\langle n,N\rangle \in \overline{\mathsf{w}}^{ws}(l)} N \supseteq M$$

since otherwise the formula $\bigwedge_{l \to r \in \mathcal{R}} [\![l \gtrsim_{\mathsf{WPO}(\mathcal{MS}um)} r]\!]$ is trivially unsatisfiable. Hence in our implementation, we consider $\mathcal{MS}um$ and $\mathcal{MP}ol$ are induced by the weight status which minimizes the number of f with $ws(f) = \mathsf{max}$, while satisfying the above condition.

6.7.3 Reducing Recursive Checks

Encoding KBO as a reduction order [91] is notably efficient, because KBO does not have a recursive check like LPO or WPO. For WPO, we can reduce formulas for recursive checks by restricting $w_0 > 0$, since under this restriction, $f(s_1, \ldots, s_n) >_{\mathcal{MP}ol} s_i$ whenever $n \geq 2$ and $ws(f) = \mathsf{pol}$. Hence when $n \geq 2$ and $ws(f) = \mathsf{pol}$, we reduce the formula $f(s_1, \ldots, s_n) \ {}_{(\sim)}\!\!\gtrsim_1 t$ to $f(s_1, \ldots, s_n) \ {}_{(\sim)}\!\!\gtrsim_2 t$. Analogously, when $m \geq 2$ and $ws(g) = \mathsf{pol}$ we reduce the formula $f(s_1, \ldots, s_n) \ {}_{(\sim)}\!\!\gtrsim_2 g(t_1, \ldots, t_m)$ to the following:

$$\mathsf{p}_f > \mathsf{p}_g \vee \mathsf{p}_f = \mathsf{p}_g \wedge [\![[s_1, \ldots, s_n]^{\sigma(f)} \gtrsim^{\mathsf{lex}}_{(\sim)\mathsf{WPO}(\mathcal{A},\sigma)} [t_1, \ldots, t_m]^{\sigma(g)}]\!]$$

without generating formulas for recursive checks corresponding to cases (2a) and (2b). Note that however, this simplification does not apply when encoding reduction pairs using argument filtering, as we will see in Section 7.5.2.

Chapter 7

Nagoya Termination Tool

In this chapter, we describe the *Nagoya Termination Tool* (NaTT), a new termination prover for TRS, which is available at

`http://www.trs.cm.is.nagoya-u.ac.jp/NaTT/`

NaTT is powerful and fast; its power comes from the implementation of WPO introduced in Chapters 4 and 5, and its efficiency comes from the strong cooperation with state-of-the-art SMT solvers. In principle, any solver that complies with the SMT-LIB Standard[1] version 2.0 can be incorporated as a back-end into NaTT.

In the next section, we present some existing DP processors that are implemented in NaTT. Section 7.2 describes the implementation of WPO and demonstrates how to obtain other existing techniques as instances of WPO. Some techniques on cooperating with SMT solvers are presented in Section 7.3. After giving some design details in Section 7.4, we verify the power of WPO through experiments in Section 7.5. We assess NaTT by its result in the *Termination Competition* [75] in Section 7.6.

7.1 Implementation of the DP Framework

The overall procedure of NaTT is illustrated in Figure 7.1.

NaTT is based on the DP framework, a very successful technique for proving termination of TRSs which is implemented in almost all the modern termination provers for TRSs. The DP framework (dis)proves termination of $\mathcal{R}$ by simplifying

[1]`http://www.smtlib.org/`

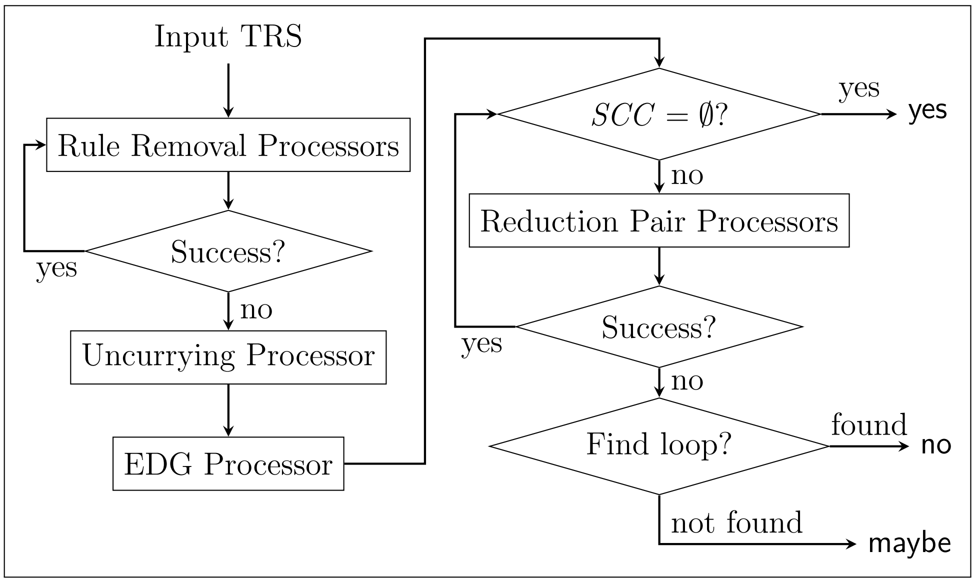


Figure 7.1: Flowchart of NaTT

and decomposing *DP problems* $\langle \mathcal{P}, \mathcal{R} \rangle$, where initially $\mathcal{P} = \mathsf{DP}(\mathcal{R})$. To this end, many *DP processors* have been proposed. NaTT implements the following DP processors:

Dependency Graph Processor

This processor decomposes a DP problem $\langle \mathcal{P}, \mathcal{R} \rangle$ into the subproblems $\langle \mathcal{P}_1, \mathcal{R} \rangle, \ldots, \langle \mathcal{P}_n, \mathcal{R} \rangle$, where $\mathcal{P}_1, \ldots, \mathcal{P}_n$ are the *strongly connected components (SCCs)* of the *dependency graph* [34, 27]. Since the dependency graph is not computable in general, several approximations called *estimated dependency graphs (EDGs)* have been proposed. NaTT implements the EDG proposed by Giesl et al. [28].

Reduction Pair Processor

This processor forms the core of NaTT; it supports the following ones:

- Some *simplification orders* combined with *argument filters* [2]:
  - KBO [43] and its variants including KBO with *status* [69], GKBO [62] and TKBO [58, 81],
  - MPO [14] and LPO [40],

- POLO [53, 2] and its variants, including certain forms[2] of *POLO* with negative constants [35] and *max-POLO* [24],
- the matrix interpretation method [38, 20], and
- WPO.

Note that all of the above mentioned reduction pairs are subsumed by WPO. That is, by implementing WPO, we obtain the other reduction pairs for free. We discuss the implementation details in Section 7.2.

Rule Removal Processor

In the worst case, the size of dependency pairs is quadratic in the size of the input TRS $\mathcal{R}$. Hence, it is preferable to reduce the size of $\mathcal{R}$ before computing dependency pairs. To this end, NaTT applies the *rule removal processor* [27]. If all rules in $\mathcal{R}$ are weakly decreasing with respect to a *monotone* reduction pair, then the processor removes strictly decreasing rules from $\mathcal{R}$. The required monotonicity of a reduction pair is obtained by choosing appropriate parameters for the implementation of WPO described above.

Uncurrying Processor

Use of uncurrying for proving termination is proposed for *applicative* rewrite systems by Hirokawa et al. [37]. The uncurrying implemented in NaTT is similar to the generalized version proposed by Sternagel and Thiemann [71], in the sense that it does not assume *application symbols* to be binary. A symbol f is considered as an application symbol if all the following conditions hold:

- f is defined and has positive arity,
- a subterm of the form $f(x, \dots)$ does not occur in any left-hand-sides of $\mathcal{R}$,
- a subterm of the form $f(g(\dots), \dots)$ occurs in some right-hand-side of $\mathcal{R}$.

If such application symbols are found, then $\mathcal{R}$ is uncurried with respect to the uncurrying TRS $\mathcal{U}$ that consists of the following rules:[3]

$$f(f^l g(x_1, \dots, x_m), y_1, \dots, y_n) \to f^{l+1} g(x_1, \dots, x_m, y_1, \dots, y_n)$$

[2]Here, negative values are allowed only for constant part.

[3]The notation is derived from the *freezing* technique [84].

for every $g \neq f$ and l less than the *applicative arity*[4] of g, where $f^0 g$ denotes g and $f^{l+1} g$ is a new function symbol of arity $m + n$.

7.2 Implementation of WPO

As we already mentioned, NaTT implements only WPO for obtaining reduction pairs. Following options are provided for specifying search spaces of parameters for WPO.

Classes of Precedences

NaTT offers "quasi" and "strict" precedences, as well as an option to disable them (i.e., all symbols are considered to have the same precedence). For reduction pairs using precedences, we recommend quasi-precedences which are chosen by default, as the encoding follows the technique of Zankl et al. [91] that naturally encodes quasi-precedences.

Classes of Status Functions

NaTT offers three classes of status functions: "total", "partial" and "empty" ones.

Templates for Weight Algebras

One of the most important tasks in proving termination by WPO is finding an appropriate weight algebra. In order to reduce the task to an SMT problem, NaTT considers *template algebras* on integer domain. Currently the following template algebras are implemented.

- The algebra $\mathcal{Pol}$ of Definition 3.14, where interpretations are in the following shape:
$$f_{\mathcal{Pol}}(x_1, \ldots, x_n) = w_f + \sum_{i=1}^{n} c_{f,i} \cdot x_i \tag{3.2}$$
where the *template variables* w_f and $c_{f,1}, \ldots, c_{f,n}$ should be decided by an external SMT solver.

[4]Applicative arities are taken so that *η-saturation* is not needed.

- The algebra $\mathcal{M}ax$ where we supply coefficients compared to Definition 4.15, in order to treat uniformly with $\mathcal{P}ol$. A symbol f with arity ≥ 1 is interpreted in the following shape:
$$f_{\mathcal{M}ax}(x_1, \dots, x_n) = \max_{i=1}^{n}(p_{f,i} + c_{f,i} \cdot x_i) \tag{7.1}$$
where $p_{f,1}, \dots, p_{f,n}$ and $c_{f,1}, \dots, c_{f,n}$ are template variables. For constant symbols, interpretations of the shape (3.2) are used. Since the operator max is not usually supported by SMT solvers, these interpretations are encoded to quantifier-free formula using the technique presented in Section 6.3.
- The algebra $\mathcal{MP}ol$ combines both form of interpretations according to the heuristics presented in Section 6.7.2.

The template variables introduced above are partitioned into two groups: the template variables $w_f, p_{f,1}, \dots, p_{f,n}$ are grouped in the *constant part*, and template variables $c_{f,1}, \dots, c_{f,n}$ are in the *coefficient part*. For efficiency, it is important to properly restrict the range of these variables.

7.2.1 Obtaining Well-known Reduction Pairs

Although most of the existing reduction pairs are subsumed by WPO, some of them are still useful for improving efficiency, due to the restricted search space and simplified SMT encoding. We list parameters that corresponds to some known reduction pairs in Tables 7.1 and 7.2. Note here that the effects of non-collapsing argument filters are simulated by allowing 0-coefficients in the weight algebra. Thus, NaTT has a dedicated implementation only for *collapsing* argument filters, and implementations of usable rules for interpretation methods and path orders are smoothly unified.

7.3 Cooperation with SMT Solvers

NaTT is designed to work with any SMT-LIB 2.0 compliant solvers that support at least `QF_LIA` logic, for which various efficient solvers exist.[5] NaTT extensively uses SMT encoding techniques for finding appropriate reduction pairs; the conditions of reduction pair processors are encoded into the following SMT constraint:

$$\bigwedge_{l \to r \in \mathcal{R}} [\![l \gtrsim r]\!] \wedge \bigwedge_{s \to t \in \mathcal{P}} [\![s \gtrsim t]\!] \wedge \bigvee_{s \to t \in \mathcal{P}} [\![s \succ t]\!] \tag{7.2}$$

[5]Confer the Satisfiability Modulo Theories Competition, `http://smtcomp.org/`.

Table 7.1: Parameters for some monotone reduction pairs.

Technique	template	coefficient	constant	precedence	status
Linear POLO	$\mathcal{P}ol$	$\mathbb{Z}_+$	$\mathbb{N}$	no	empty
LPO	$\mathcal{M}ax$	$\{1\}$	$\{0\}$	yes	total
KBO[6]	$\mathcal{P}ol$	$\{1\}$	$\mathbb{N}$	yes	total
Transfinite KBO[6]	$\mathcal{P}ol$	$\mathbb{Z}_+$	$\mathbb{N}$	yes	total

Table 7.2: Parameters for some (non-monotone) reduction pairs.

Technique	template	coefficient	constant	precedence	status
Linear POLO	$\mathcal{P}ol$	$\mathbb{N}$	$\mathbb{N}$	no	empty
Max-POLO	$\mathcal{MP}ol$	$\mathbb{N}$	$\mathbb{Z}$	no	empty
LPO + argument filter	$\mathcal{M}ax$	$\{0,1\}$	$\{0\}$	yes	total
KBO + argument filter	$\mathcal{P}ol$	$\{0,1\}$	$\mathbb{N}$	yes	total
Matrix interpretations	$\mathcal{P}ol$	$\mathbb{N}^{d\times d}$	$\mathbb{N}^d$	no	empty
WPO($\mathcal{MS}um$)	$\mathcal{MP}ol$	$\{0,1\}$	$\mathbb{N}$	yes	partial

where each $[\![l \underset{(\sim)}{>} r]\!]$ is an SMT formula that represents the condition $l \underset{(\sim)}{>} r$. In the remainder of this section, we present two techniques for handling such constraints that contribute to the efficiency of NaTT.

7.3.1 Use of Interactive Features of SMT Solvers

In a typical run of termination verification, constraints of the form (7.2) are generated and solved many times, and each encoding sometimes involves thousands of lines of SMT queries with a number of template and auxiliary variables. Hence, runtime spent for the SMT solver forms a large part of the overall runtime of the tool execution. NaTT tries to reduce the runtime by using *interactive* features of SMT solvers, which are specified in SMT-LIB 2.0.

Note that for each technique of reduction pairs, the encoded formula of the constraint $\bigwedge_{l\to r\in\mathcal{R}}[\![l \gtrsim r]\!]$ in (7.2) need not be changed during a run, as far as $\mathcal{R}$ is not modified.[7] Hence, when a reduction pair processor is applied for the first time, the back-end SMT solver is initialized according to the following pseudo-script:

[6]Further constraints for *admissibility* are imposed.

[7]Although rules in $\mathcal{R}$ may be removed by considering *usable rules*, the formula still need not be changed, since it can be simulated by negating a propositional variable that represents whether the rule is usable or not.

(assert ($\bigwedge_{l \to r \in \mathcal{R}}(\mathsf{u}_{l \to r} \Rightarrow [\![l \gtrsim r]\!])$))
(push)

where $\mathsf{u}_{l \to r}$ is a boolean variable denoting whether the rule $l \to r$ is usable or not. When the processor is applied to an SCC $\mathcal{P}$, the following script is used:

(assert ($\bigwedge_{s \to t \in \mathcal{P}}[\![s \gtrsim t]\!] \wedge \bigvee_{s \to t \in \mathcal{P}}[\![s \succ t]\!]$))
(check-sat)

Then, if a solution is found by the SMT solver, NaTT analyzes the solution using the `get-value` command. After this analysis, the command

```
(pop)
```

is issued to clear the constraints due to $\mathcal{P}$ and go back to the context saved by the `(push)` command. In order to derive the best performance of the solver,

```
(reset)
```

is also issued in case sufficiently many rules become unusable (e.g., 1/3 of the rules in $\mathcal{R}$) from $\mathcal{P}$. All these commands, `push`, `pop` and `reset` are expected to be available in SMT-LIB 2.0 compliant solvers.

7.3.2 Use of Linear Arithmetic

Expressions of the form (3.2) or (7.1) are nonlinear, due to the coefficients $c_{f,1}, \dots, c_{f,n}$. However, not many SMT solvers support *nonlinear* arithmetic, and even if they do, they are much less scalable than they are for linear arithmetic. Hence, we consider reducing the formulas to linear ones by restricting the range of $c_{f,1}, \dots, c_{f,n}$ e.g. to $\{0, 1\}$. Although the idea is inspired by the technique of Borralleras et al. [10], NaTT uses a more straightforward reduction using `ite` (*if-then-else*) expressions. Each coefficient $c_{f,i}$ is replaced by the expression (ite $b_{f,i}$ 1 0) where $b_{f,i}$ is a propositional variable, and then multiplications are reduced according to the rule:

(* (ite e_1 e_2 e_3) e_4) $\to$ (ite e_1 (* e_2 e_4) (* e_3 e_4))

It is easy to see that this reduction terminates and linearizes expressions of the form (3.2) or (7.1). Moreover, it is possible to avoid an explosion of the size of formulas by introducing an auxiliary variable for the duplicated expression e_4.

Example 7.1 *Consider the constraint* $\mathsf{f}(\mathsf{f}(\mathsf{a})) > \mathsf{b}$ *interpreted in the algebra* $\mathcal{P}ol$*, and suppose that the range of* $c_{\mathsf{f},1}$ *is restricted to* $\{1, 2\}$*. The interpretation of the term* $\mathsf{f}(\mathsf{f}(\mathsf{a}))$ *is reduced as follows (written as S-expressions):*

$$\begin{aligned} [\![\mathsf{f}(\mathsf{f}(\mathsf{a}))]\!] &= \texttt{(+ } w_{\mathsf{f}} \texttt{ (* (ite } b_{\mathsf{f},1} \texttt{ 2 1) } [\![\mathsf{f}(\mathsf{a})]\!]\texttt{))} \\ &\to \texttt{(+ } w_{\mathsf{f}} \texttt{ (ite } b_{\mathsf{f},1} \texttt{ (* 2 } [\![\mathsf{f}(\mathsf{a})]\!]\texttt{) } [\![\mathsf{f}(\mathsf{a})]\!]\texttt{))} \end{aligned}$$

Similarly, for $\mathsf{f}(\mathsf{a})$ *we obtain*

$$[\![\mathsf{f}(\mathsf{a})]\!] \to \texttt{(+ } w_{\mathsf{f}} \texttt{ (ite } b_{\mathsf{f},1} \texttt{ (* 2 } w_{\mathsf{a}}\texttt{) } w_{\mathsf{a}}\texttt{))}$$

Now, the constraint $[\![\mathsf{f}(\mathsf{f}(\mathsf{a})) > \mathsf{b}]\!]$ *is expressed by the following script:*

$$\begin{aligned} &\texttt{(define-fun } v \texttt{ (+ } w_{\mathsf{f}} \texttt{ (ite } b_{\mathsf{f},1} \texttt{ (* 2 } w_{\mathsf{a}}\texttt{) } w_{\mathsf{a}}\texttt{)))} \\ &\texttt{(assert (> (+ } w_{\mathsf{f}} \texttt{ (ite } b_{\mathsf{f},1} \texttt{ (* 2 } v\texttt{) } v\texttt{) } w_{\mathsf{b}}\texttt{)))} \end{aligned}$$

In contrast to SAT encoding techniques [23, 24, 20], we do not have to care about the bit-width for the constant part and intermediate results. It is also possible to indicate that NaTT should keep formulas nonlinear, and solve them using SMT solvers that support QF_NIA logic. Our experiments, however, suggest that use of nonlinear SMT solving is much inefficient and not beneficial on the problems from TPDB.

7.4 Design

The source code of NaTT consists of about 6000 lines of code written in OCaml.[8] About 23% is consumed by interfacing SMT solvers, where some optimizations for encodings are also implemented. Another 17% is for parsing command-lines and TRS files. The most important part of the source code is the 40% devoted to the implementation of WPO, the unified reduction pair processor. Each of the other processors implemented consumes less than 3%. For computing SCCs, the third-party library `ocamlgraph`[9] is used.

7.4.1 Command Line Interface

The command line of NaTT has the following syntax:

[8] http://caml.inria.fr/

[9] http://ocamlgraph.lri.fr/

```
./NaTT [FILE] [OPTION]...  [PROCESSOR]...
```

To execute `NaTT`, an SMT-LIB 2.0 compliant solver must be installed. By default, `z3` version 4.0 or later[10] is supposed to be installed in the path. Users can specify other solvers by the `--smt "COMMAND"` option, where the solver invoked by `COMMAND` should process SMT-LIB 2.0 scripts given on the standard input.

The TRS whose termination should be verified is read from either the specified `FILE` or the standard input.[11] Each `PROCESSOR` is either an order (e.g. `POLO`, `KBO`, `WPO`, *etc.*, possibly followed by options), or a name of other processors (`UNCURRY`, `EDG`, or `LOOP`). Orders preceding the `EDG` processor should be monotone reduction pairs and applied as rule removal processors before computing the dependency pairs. Orders following the `EDG` processor are applied as reduction pair processors to each SCC in the EDG. A list of available `OPTION`s and `PROCESSOR`s can be obtained via `NaTT --help`.

7.4.2 The Default Strategy

In case no `PROCESSOR` is specified, the following default strategy will be applied:

- As a rule removal processor, POLO with coefficients in $\{1, 2\}$ and constants in $\mathbb{N}$ is applied.
- Then the uncurrying processor is applied.
- The following reduction pair processors are applied (in this order):
  1. POLO with coefficients in $\{0, 1\}$ and constants in $\mathbb{N}$,
  2. algebra $\mathcal{M}ax$ with coefficients in $\{0, 1\}$ and constants in $\mathbb{N}$,
  3. LPO with quasi-precedence, status and argument filters,
  4. algebra $\mathcal{MP}ol$ with coefficients in $\{0, 1\}$ and constants in $\mathbb{Z}$,
  5. WPO with quasi-precedence, partial status, algebra $\mathcal{MP}ol$, coefficients in $\{0, 1\}$ and constants in $\mathbb{N}$,
  6. matrix interpretations with $\{0, 1\}^{2\times2}$ matrices and $\mathbb{N}^2$ vectors.
- If all the above processors fail, then a (naive) loop detection is performed.

[10] `http://z3.codeplex.com/`

[11] The format is found at `https://www.lri.fr/~marche/tpdb/format.html`.

7.5 Experiments

In this section, we examine the performance of WPO implemented in NaTT, both as a reduction order and reduction pairs.

For a test set of termination problems, we use the 1463 TRSs from the TRS Standard category of TPDB 8.0.6 [77]. The experiments are run on a server equipped with two quad-core Intel Xeon W5590 processors running at a clock rate of 3.33GHz and 48GB of main memory, though only one thread of SMT solver runs at once. As the SMT solver, we choose z3 4.3.1.[12] Timeout is set to 60s, as in the Termination Competition.

7.5.1 Results for Reduction Orders

First we evaluated WPO as a reduction order by directly testing orientability for input TRSs. The results are listed in Table 7.3. Since KBO, POLO($\mathcal{S}um$), WPO($\mathcal{S}um$) are only applicable for non-duplicating TRSs, the test set is split into non-duplicating ones (consisting of 439 TRSs) and duplicating ones (consisting of 1024 TRSs). In the table, 'yes' column indicates the number of successful termination proofs, 'T.O.' indicates the number of timeouts, and 'time' indicates the total time. We point that WPO($\mathcal{MS}um$) is a balanced choice; it is significantly stronger than existing orders, while the runtime is much better than involving non-linear SMT solving (last 5 columns). Note that here we directly solve non-linear problems using z3; their runtime will be significantly improved by the linearization proposed in Section 7.3.2. If the efficiency is the main concern, then WPO($\mathcal{S}um^+$), a variant of WPO($\mathcal{S}um$) with $w_0 > 0$, is a reasonable substitute for KBO. This efficiency is due to the reduction of recursive checks proposed in Section 6.7.3.

7.5.2 Results for Reduction Pairs

Second, we evaluated WPO as a reduction pair. Table 7.4 compares the power of the reduction pair processors. In 'total status' column, we apply standard total status and argument filtering to obtain a reduction pair from a reduction order, as in [85]. Because of argument filtering, existence of duplicating rules are not an issue in this setting. For POLO, status is ignored.

[12] http://z3.codeplex.com/

Table 7.3: Results for reduction orders.

		439 non-dup. TRSs			1024 dup. TRSs		
order	algebra	yes	T.O.	time	yes	T.O.	time
POLO	$\mathcal{S}um$	41	0	3.54	–	–	–
LPO		90	0	20.94	90	0	22.64
KBO		115	0	4.51	–	–	–
WPO	$\mathcal{S}um^+$	126	0	4.75	–	–	–
WPO	$\mathcal{S}um$	135	0	31.47	–	–	–
WPO	$\mathcal{MS}um$	135	0	31.55	138	0	40.06
WPO	$\mathcal{M}ax$	116	0	52.96	125	0	35.23
POLO	$\mathcal{P}ol$	81	3	190.45	12	9	812.18
TKBO		125	0	10.38	22	9	1146.65
POLO	$\mathcal{M}at$	136	3	246.73	20	15	1638.92
WPO	$\mathcal{P}ol$	147	0	276.21	26	9	1261.37
WPO	$\mathcal{MP}ol$	147	0	276.05	138	9	982.71

The power of WPO is still measurable here. On the contrary to the reduction order case, WPO($\mathcal{S}um$) outperforms KBO both in power and efficiency. This is because KBO needs formulas for recursive comparison that resembles WPO, when argument filters are considered. Moreover, encodings of weights are more complex in KBO, since w_0 cannot be fixed to 0 as discussed in Section 6.7.1. Finally, KBO needs extra constraints that correspond to admissibility.

In 'partial status' column, we moreover admit partial status functions. We also apply partial status for KBO [86] but not for LPO; note that LPO does not benefit from partial status because weights are not considered. The power of WPO is much more significant in this setting, and WPO($\mathcal{MS}um$) is about 30% stronger than any other existing techniques. Though the efficiency is sacrificed for partial status, this is not a severe problem in the DP framework, as we will see in the next section. On the other hand, our implementation of the instances of WPO that require non-linear SMT solving are extremely time-consuming. Especially, WPO($\mathcal{M}at$) looses 107 problems by timeout compared to the standard matrix interpretation method. We conjecture that the situation can be improved by SAT encoding or by using other non-linear SMT solvers such as the techniques of Zankl and Middeldorp [90] and Borralleras et al. [11].

Table 7.4: Results for reduction pairs.

		total status			partial status		
order	algebra	yes	T.O.	time	yes	T.O.	time
POLO	$\mathcal{S}um$	512	0	111.51	–	–	–
POLO	$\mathcal{MS}um$	522	0	273.09	–	–	–
LPO		502	0	361.41	–	–	–
KBO		497	3	900.79	520	4	1109.59
WPO	$\mathcal{S}um$	514	23	830.21	560	4	1105.42
WPO	$\mathcal{M}ax$	548	6	1230.17	637	13	1855.59
WPO	$\mathcal{MS}um$	578	6	1289.61	675	12	1755.66
POLO	$\mathcal{P}ol$	544	19	1958.44	–	–	–
POLO	$\mathcal{MP}ol$	540	18	1889.86	–	–	–
POLO	$\mathcal{M}at$	645	480	32367.26	–	–	–
TKBO		516	187	15665.26	539	178	15799.28
WPO	$\mathcal{P}ol$	527	172	14535.24	579	153	13579.95
WPO	$\mathcal{MP}ol$	560	88	7678.43	672	94	9269.36
WPO	$\mathcal{M}at$	–	–	–	538	640	42067.45

7.5.3 Effect of Linearlization

In this section, we verify the effect of linearization proposed in Section 7.3.2.

Table 7.5 presents the results of linearization for reduction orders. For each order, the coefficient part is bounded by 2. In "non-linear" column, the encoded QF-NIA problem is solved via `z3` and the constant part is also bounded by 2, in order to achieve a feasible runtime. In "linearized" column the constant part is left open in $\mathbb{N}$. The effect of linearization is clear here; the runtime improves by an order of magnitude, and several successful termination proofs are gained from the unlimited constant part. The effect becomes even clearer if we increase the upper bound of the constant part for non-linear case.

Table 7.6 presents the results for reduction pairs. The upper bounds for the constant part and the coefficient part are as in the previous setting. The benefit in runtime becomes less significant but still measurable in this setting (except for matrix interpretations). If we increase the upper bound of the constant part for non-linear case, the difference becomes dramatically clearer.

Table 7.5: Effect of linearization (reduction orders).

	non-linear			linearized		
reduction order	yes	T.O.	time	yes	T.O.	time
$\mathcal{P}ol$	93	12	1002.63	122	0	29.17
$\mathcal{M}at$	156	18	1885.65	183	0	199.94
TKBO	147	12	1359.06	162	0	61.55
WPO($\mathcal{P}ol$)	173	12	1537.58	175	0	159.67
WPO($\mathcal{MP}ol$)	285	12	1258.76	289	0	212.47

Table 7.6: Effect of linearization (reduction pairs).

	non-linear			linearized		
reduction pair	yes	T.O.	time	yes	T.O.	time
$\mathcal{P}ol$	516	18	1814.31	543	6	851.01
$\mathcal{M}at$	644	13	1972.56	648	15	1999.78
TKBO	511	17	2573.66	521	11	2355.69
WPO($\mathcal{P}ol$)	578	22	2946.16	579	14	2366.34
WPO($\mathcal{MP}ol$)	668	26	3320.65	669	22	3096.16

7.5.4 Effect of Interactive SMT Solving

In Table 7.7, we verify the effect of interactive use of SMT solvers proposed in Section 7.3.1. The “interactive” column uses the interactive feature, and the “reset” column issues the `reset` command at every time the reduction pair is applied. The effect is especially clear for methods that involve recursive comparisons (LPO, KBO, and WPO). On the other hand, the benefit is unclear for interpretation methods; for these methods, encoded formulas are simple arithmetic comparisons which are easy to generate, and the overhead of saving contexts seems to become significant. Note that in the “reset” column, NaTT optimizes the set of considered rules by excluding unusable rules. Due to this optimization, the “reset” column gains several successes compared to the “interactive” column.

Table 7.7: Effect of interactive SMT solving.

	reset			interactive		
reduction pair	yes	T.O.	time	yes	T.O.	time
$\mathcal{S}um$	512	0	137.09	512	0	111.51
$\mathcal{M}at$	647	15	1998.84	648	17	2200.75
LPO	502	2	781.98	502	2	361.41
KBO	497	4	1778.15	496	2	900.79
WPO($\mathcal{MS}um$)	679	15	2994.93	675	12	1755.66

Table 7.8: Results for Combination

strategy	yes	no	unknown	T.O.	time
NaTT with WPO	848	173	429	13	1865.50
NaTT without WPO	810	173	467	13	2023.18
AProVE	1020	270	0	173	15123.48
T$_\mathsf{T}$T$_2$	788	193	417	65	13784.43

7.5.5 Results for Combination

Modern termination provers combine DP processors and apply weak but efficient ones first. In Table 7.8, we compare the following: 'NaTT with WPO' (the default strategy described in Section 7.4.2), 'NaTT without WPO' (WPO is replaced by KBO),[13] AProVE 2014, and T$_\mathsf{T}$T$_2$ 1.15. The 'no' column indicates the number of successful nontermination proofs.

The benefit of WPO is clear in this setting. Not only gaining 38 successful termination proofs, WPO also advances the efficiency of the tool. The reason for this phenomenon is that WPO may prove termination of a TRS, which otherwise would be passed to the more inefficient matrix interpretation or nontermination check which will never succeed. When compared to other termination provers, NaTT is not as strong as AProVE. This is because AProVE implements lots of techniques that are not implemented in NaTT. Nonetheless, NaTT (dis)proves 37 examples that could not be proved by AProVE or T$_\mathsf{T}$T$_2$. Moreover, NaTT is extremely fast than other tools.

[13]However, KBO does not contribute in this strategy.

7.6 Assessment

Many tools have been developed for proving termination of TRSs, and the international Termination Competition has been held annually for a decade. NaTT participated in the *TRS Standard* category of the full-run 2013,[14] where the other participants are versions of: AProVE,[15] $\mathsf{T_TT_2}$,[16] MU-TERM,[17] and WANDA.[18] Using the default strategy described in Section 7.4.2, NaTT (dis)proves termination of 982 TRSs out of 1463 TRSs,[19] and comes next to (the two versions of) AProVE, the constant champion of the category. It should be noticed that NaTT proved termination of 34 TRSs out of the 159 whose termination could not be proved by any other tool, and for 29 of them, WPO is essential. NaTT is also notably faster than the other competitors; it consumed only 21% of the time compared to AProVE, the second fastest. We expect that we can further improve efficiency by optimizing to multi-core architecture; currently, NaTT runs in almost single thread.

NaTT also participated in the *SRS Standard* category. However, the result is not as good as it is for TRSs. This is due to the fact that the default strategy of Section 7.4.2 is designed only for non-unary signatures. Indeed, an interpretation of the form (7.1) makes sense only if it has at least two arguments. It should be improved by choosing a strategy depending on the shape of input TRSs.

[14]`http://termcomp.uibk.ac.at/`

[15]`http://aprove.informatik.rwth-aachen.de/`

[16]`http://cl-informatik.uibk.ac.at/software/ttt2/`

[17]`http://zenon.dsic.upv.es/muterm/`

[18]`http://wandahot.sourceforge.net/`

[19]due to an unfortunate parser bug, the competition version of NaTT failed to input 36 TRSs for which current version succeeds to conclude termination.

Chapter 8

AC-Compatible Knuth-Bendix Orders

Associative and commutative (AC) operators appear in many applications, e.g. in automated reasoning with respect to algebraic structures such as commutative groups or rings. AC termination is important when deciding validity in equational theories with AC operators by means of completion.

Several termination methods for plain rewriting have been extended to deal with AC symbols.

- Lankford [55] presented some restrictions of polynomial interpretations that ensure compatiblility with AC axioms. Ben Cherifa and Lescanne [7] gave a sound and complete characterization of such polynomial interpretations.
- There have been numerous papers on extending RPO of Dershowitz [14] to deal with AC symbols. To list some, [18, 5, 33, 42, 4, 66, 41], and culminating in the fully syntactic AC-RPO of Rubio [65].
- Several authors, namely, Kusakari and Toyama [51], Marché and Urbain [60], Giesl and Kapur [26] adapted the influential dependency pair method of Arts and Giesl [2] to AC rewriting. The AC extension of the more modern DP framework is due to Alarcón et al. [1].

On the other hand, we are aware of only two papers on AC extensions of the order of Knuth and Bendix (KBO) [43]. In this chapter we revisit these orders and present yet another AC-compatible KBO. Steinbach [70] presented a first version, which comes with the restriction that AC symbols are minimal in the precedence. By incorporating ideas of AC-RPO [65], Korovin and Voronkov [46] presented a version without this restriction. Actually, they present two versions. One is defined on ground terms and another one on arbitrary terms. For (automatically) proving AC termination of rewrite systems, an AC-compatible order on arbitrary terms is required.[1] We show that the second order of Korovin and Voronkov [46] lacks the monotonicity property which is required by the definition of simplification orders. Nevertheless we prove that the order is sound for proving termination by extending it to an AC-compatible simplification order. We furthermore present a simpler variant of this latter order which properly extends the order of Steinbach [70]. In particular, Steinbach's order is a correct AC-compatible simplification order, contrary to what is claimed in [46]. We also present new complexity results which confirm that AC rewriting is much more involved than plain rewriting. Apart from these theoretical contributions, we implemented the various AC-compatible KBOs to compare them also experimentally.

The remainder of this chapter is organized as follows. After recalling basic concepts of rewriting modulo AC and orders, we revisit Steinbach's order in Section 8.2. Section 8.3 is devoted to the two orders of Korovin and Voronkov. We present a first version of our AC-compatible KBO in Section 8.4 and theoretically compare the order with other variants introduced so far. In Section 8.5, we give the non-trivial proofs that our AC-KBO as well as the corrected version of Korovin and Voronkov's order have the required properties. (The proofs in [46] are limited to the order on ground terms.) Then we investigate complexities of checking membership and orientability in Section 8.6 for the various orders. In Section 8.7 our order is strengthened with subterm coefficients. In order to show effectiveness of these orders, experimental data is provided in Section 8.8.

8.1 Rewriting modulo AC

In this section, we briefly recall the notions of AC-rewriting. We consider a designated subset $\mathcal{F}_{\mathsf{AC}} \subseteq \mathcal{F}$ of binary AC symbols.

[1] Any AC-compatible reduction order $\succ_{\mathrm{g}}$ on ground terms can trivially be extended to arbitrary terms by defining $s \succ t$ iff $s\sigma \succ_{\mathrm{g}} t\sigma$ for all grounding substitutions σ. This is, however, only of (mild) theoretical interest.

Definition 8.1 *The TRS* AC *consists of the following rules for every* $f \in \mathcal{F}_{\mathsf{AC}}$*:*

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

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

The AC-equivalence *relation is defined as* $\approx_{\mathsf{AC}} := (\xrightarrow[\mathsf{AC}]{} \cup \xleftarrow[\mathsf{AC}]{})^*$.

Definition 8.2 *Let* $\mathcal{R}$ *be a TRS. The* AC-rewrite relation *induced by* $\mathcal{R}$ *is defined as:* $\xrightarrow[\mathcal{R}/\mathsf{AC}]{} := \approx_{\mathsf{AC}} \cdot \xrightarrow[\mathcal{R}]{} \cdot \approx_{\mathsf{AC}}$. *The TRS* $\mathcal{R}$ *is* AC-terminating *iff the relation* $\approx_{\mathsf{AC}} \cdot \xrightarrow[\mathcal{R}]{} \cdot \approx_{\mathsf{AC}}$ *is well-founded.*

Definition 8.3 *A strict order* $\succ$ *on terms is* AC-compatible *iff* $\approx_{\mathsf{AC}} \cdot \succ \cdot \approx_{\mathsf{AC}} \subseteq \succ$, *and is* AC-total *iff* $s \succ t$, $t \succ s$ *or* $s \approx_{\mathsf{AC}} t$, *for all ground terms* s *and* t.

Note that AC compatibility can be rephraised by that $\langle \approx_{\mathsf{AC}}, \succ \rangle$ is an order pair. AC termination is established if a TRS is oriented by an *AC-compatible simplification order.*

8.2 Steinbach's Order

In this section, we recall the AC-compatible KBO $>_{\mathsf{S}}$ of Steinbach [70], which reduces to the standard KBO if AC symbols are absent.[2] Just like the standard KBO, $>_{\mathsf{S}}$ depends on a precedence and an admissible weight function. In this chapter, we only consider a strict precedence.

Definition 8.4 *The* top-flattening *[65] of a term* t *with respect to an AC symbol* f *is the multiset* $\nabla_f(t)$ *defined inductively as follows:* $\nabla_f(t) = \{t\}$ *if* $\mathsf{root}(t) \neq f$ *and* $\nabla_f(f(t_1,t_2)) = \nabla_f(t_1) \uplus \nabla_f(t_2)$.

Definition 8.5 *Let* $>$ *be a precedence and* $\langle w, w_0 \rangle$ *a weight function. The order* $>_{\mathsf{S}}$ *is inductively defined as follows:* $s >_{\mathsf{S}} t$ *if* $|s|_x \geq |t|_x$ *for all* $x \in \mathcal{V}$ *and either* $w(s) > w(t)$, *or* $w(s) = w(t)$ *and one of the following alternatives holds:*

0. $s = f^k(t)$ *and* $t \in \mathcal{V}$ *for some* $k > 0$,

1. $s = f(s_1,\dots,s_n)$, $t = g(t_1,\dots,t_m)$, *and* $f > g$,

2. $s = f(s_1,\dots,s_n)$, $t = f(t_1,\dots,t_n)$, $f \notin \mathcal{F}_{\mathsf{AC}}$, $[s_1,\dots,s_n] >^{\mathsf{lex}}_{\mathsf{S}} [t_1,\dots,t_n]$,

[2]The version in [70] is slightly more general, since non-AC function symbols can have arbitrary status. To simplify the discussion, we do not consider status in this chapter.

3. $s = f(s_1, s_2)$*,* $t = f(t_1, t_2)$*,* $f \in \mathcal{F}_{\mathsf{AC}}$*, and* $\nabla_f(s) >_{\mathsf{S}}^{\mathsf{mul}} \nabla_f(t)$*.*

The relation $\approx_{\mathsf{AC}}$ *is used as preorder in* $>_{\mathsf{S}}^{\mathsf{lex}}$ *and* $>_{\mathsf{S}}^{\mathsf{mul}}$*.*

Cases 0–2 are the same as in the classical Knuth-Bendix order. In case 3 terms rooted by the same AC symbol f are treated by comparing their top-flattenings in the multiset extension of $>_{\mathsf{S}}$.

Example 8.6 *Consider the signature* $\mathcal{F} = \{\mathsf{a}, \mathsf{f}, \mathsf{g}\}$ *with* $\mathsf{f} \in \mathcal{F}_{\mathsf{AC}}$*, precedence* $\mathsf{g} > \mathsf{a} > \mathsf{f}$ *and admissible weight function* $\langle w, w_0 \rangle$ *with* $w(\mathsf{f}) = w(\mathsf{g}) = 0$ *and* $w_0 = w(\mathsf{a}) = 1$*. Let* $\mathcal{R}_1$ *be the following ground TRS:*

$$\ell_1 = \mathsf{g}(\mathsf{f}(\mathsf{a},\mathsf{a})) \to \mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})) = r_1 \tag{8.1}$$

$$\ell_2 = \mathsf{f}(\mathsf{a},\mathsf{g}(\mathsf{g}(\mathsf{a}))) \to \mathsf{f}(\mathsf{g}(\mathsf{a}),\mathsf{g}(\mathsf{a})) = r_2 \tag{8.2}$$

For $i \in \{1, 2\}$*, let* $S_i = \nabla_{\mathsf{f}}(\ell_i)$ *and* $T_i = \nabla_{\mathsf{f}}(r_i)$*. Both rules vacuously satisfy the variable condition. We have* $w(\ell_1) = 2 = w(r_1)$ *and* $\mathsf{g} > \mathsf{f}$*, so* $\ell_1 >_{\mathsf{S}} r_1$ *holds by case 1. We have* $w(\ell_2) = 2 = w(r_2)$*,* $S_2 = \{\mathsf{a}, \mathsf{g}(\mathsf{g}(\mathsf{a}))\}$*, and* $T_2 = \{\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})\}$*. Since* $\mathsf{g}(\mathsf{a}) >_{\mathsf{S}} \mathsf{a}$ *holds by case 1,* $\mathsf{g}(\mathsf{g}(\mathsf{a})) >_{\mathsf{S}} \mathsf{g}(\mathsf{a})$ *holds by case 2, and therefore* $\ell_2 >_{\mathsf{S}} r_2$ *by case 3.*

Theorem 8.7 ([70]) *If every symbol in* $\mathcal{F}_{\mathsf{AC}}$ *is minimal with respect to* $>$*, then* $>_{\mathsf{S}}$ *is an AC-compatible simplification order.*[3] $\square$

In Section 8.4 we reprove[4] Theorem 8.7 by showing that $>_{\mathsf{S}}$ is a special case of our new AC-compatible Knuth-Bendix order.

8.3 Korovin and Voronkov's Orders

In this section we recall the orders of Korovin and Voronkov [46].

8.3.1 Ground Case

The first one is defined on ground terms. The difference with $>_{\mathsf{S}}$ is that in case 3 of the definition a further case analysis is performed based on terms in

[3]In [70] AC symbols are further required to have weight 0 because terms are flattened. Our version of $>_{\mathsf{S}}$ does not impose this restriction due to the use of top-flattening.

[4]The counterexample in [46] against the monotonicity of $>_{\mathsf{S}}$ is invalid as the condition that AC symbols are *minimal* in the precedence is not satisfied.

$\nabla_f(s)$ and $\nabla_f(t)$ whose root symbols are not smaller than f in the precedence. Rather than recursively comparing these terms with the order being defined, a lighter non-recursive version is used in which the weights and root symbols are considered.

Definition 8.8 *Given a multiset T of terms, a function symbol f, and a binary relation $\sqsupset$ on function symbols, we define the following submultisets of T:*

$$T{\restriction_{\mathcal{V}}} = T \cap \mathcal{V} \qquad\qquad T{\restriction_f^{\sqsupset}} = \{t \in T \setminus \mathcal{V} \mid \mathsf{root}(t) \sqsupset f\}$$

Definition 8.9 *Let $>$ be a precedence and $\langle w, w_0\rangle$ a weight function.*[5] *First we define the auxiliary relations $=_{\mathsf{kv}}$ and $>_{\mathsf{kv}}$ as follows:*

- *$s =_{\mathsf{kv}} t$ if $w(s) = w(t)$ and $\mathsf{root}(s) = \mathsf{root}(t)$,*
- *$s >_{\mathsf{kv}} t$ if either $w(s) > w(t)$ or both $w(s) = w(t)$ and $\mathsf{root}(s) > \mathsf{root}(t)$.*

The order $>_{\mathsf{KV}}$ is inductively defined on ground terms as follows: $s >_{\mathsf{KV}} t$ if either $w(s) > w(t)$, or $w(s) = w(t)$ and one of the following alternatives holds:

1. *$s = f(s_1, \ldots, s_n)$, $t = g(t_1, \ldots, t_m)$, and $f > g$,*
2. *$s = f(s_1, \ldots, s_n)$, $t = f(t_1, \ldots, t_n)$, $f \notin \mathcal{F}_{\mathsf{AC}}$, $[s_1, \ldots, s_n] >_{\mathsf{KV}}^{\mathsf{lex}} [t_1, \ldots, t_n]$,*
3. *$s = f(s_1, s_2)$, $t = f(t_1, t_2)$, $f \in \mathcal{F}_{\mathsf{AC}}$, and for $S = \nabla_f(s)$ and $T = \nabla_f(t)$*
   (a) *$S{\restriction_f^{\not<}} >_{\mathsf{kv}}^{\mathsf{mul}} T{\restriction_f^{\not<}}$, or*
   (b) *$S{\restriction_f^{\not<}} =_{\mathsf{kv}}^{\mathsf{mul}} T{\restriction_f^{\not<}}$ and $|S| > |T|$, or*
   (c) *$S{\restriction_f^{\not<}} =_{\mathsf{kv}}^{\mathsf{mul}} T{\restriction_f^{\not<}}$, $|S| = |T|$, and $S >_{\mathsf{KV}}^{\mathsf{mul}} T$.*

Here $\approx_{\mathsf{AC}}$ is used as preorder in $>_{\mathsf{KV}}^{\mathsf{lex}}$ and $>_{\mathsf{KV}}^{\mathsf{mul}}$ whereas $=_{\mathsf{kv}}$ is used in $>_{\mathsf{kv}}^{\mathsf{mul}}$.

Only in cases 2 and (3c) the order $>_{\mathsf{KV}}$ is used recursively. In case 3 terms rooted by the same AC symbol f are compared by extracting from the top-flattenings S and T the multisets $S{\restriction_f^{\not<}}$ and $T{\restriction_f^{\not<}}$ consisting of all terms rooted by a function symbol not smaller than f in the precedence. If $S{\restriction_f^{\not<}}$ is larger than $T{\restriction_f^{\not<}}$ in the multiset extension of $>_{\mathsf{kv}}$, we conclude in case (3a). Otherwise the multisets must be equal (with respect to $=_{\mathsf{kv}}^{\mathsf{mul}}$). If S has more terms than T, we conclude in case (3b). In the final case (3c), S and T have the same number of terms and we compare S and T in the multiset extension of $>_{\mathsf{KV}}$.

[5]Here we do not impose totality on precedences, cf. [46]. See also Example 8.22.

Theorem 8.10 ([46]) *The order $>_{\mathsf{KV}}$ is an AC-compatible simplification order on ground terms. If $>$ is total then $>_{\mathsf{KV}}$ is AC-total on ground terms.* □

The two orders $>_{\mathsf{KV}}$ and $>_{\mathsf{S}}$ are incomparable on ground TRSs.

Example 8.11 *Consider again the ground TRS $\mathcal{R}_1$ of Example 8.6. To orient rule* (8.1) *with $>_{\mathsf{KV}}$, the weight of the unary function symbol* g *must be* 0 *and admissibility demands* $\mathsf{g} > \mathsf{a}$ *and* $\mathsf{g} > \mathsf{f}$. *Hence rule* (8.1) *is handled by case 1 of the definition. For rule* (8.2), *the multisets*

$$S = \{\mathsf{a}, \mathsf{g}(\mathsf{g}(\mathsf{a}))\} \qquad\qquad T = \{\mathsf{g}(\mathsf{a}), \mathsf{g}(\mathsf{a})\}$$

are compared in case 3. We have $S{\restriction}_{\mathsf{f}}^{\not<} = \{\mathsf{g}(\mathsf{g}(\mathsf{a}))\}$ if $\mathsf{f} > \mathsf{a}$ and $S{\restriction}_{\mathsf{f}}^{\not<} = S$ otherwise. In both cases we have $T{\restriction}_{\mathsf{f}}^{\not<} = T$. Note that neither $\mathsf{a} >_{\mathsf{kv}} \mathsf{g}(\mathsf{a})$ nor $\mathsf{g}(\mathsf{g}(\mathsf{a})) >_{\mathsf{kv}} \mathsf{g}(\mathsf{a})$ holds. Hence case (3a) does not apply. But also cases (3b) and (3c) are not applicable as $\mathsf{g}(\mathsf{g}(\mathsf{a})) =_{\mathsf{kv}} \mathsf{g}(\mathsf{a})$ and $\mathsf{a} \neq_{\mathsf{kv}} \mathsf{g}(\mathsf{a})$. Hence, independent of the choice of $>$, $\mathcal{R}_1$ cannot be proved terminating by $>_{\mathsf{KV}}$. Conversely, the TRS $\mathcal{R}_2$ resulting from reversing rule (8.2) *in $\mathcal{R}_1$ can be proved terminating by $>_{\mathsf{KV}}$ but not by $>_{\mathsf{S}}$.*

8.3.2 Non-ground Case

Next we present the second order of Korovin and Voronkov [46], the extension of $>_{\mathsf{KV}}$ to non-ground terms. Since it coincides with $>_{\mathsf{KV}}$ on ground terms, we use the same notation for the order. In case 3 of the Definition 8.13, also variables appearing in the top-flattenings S and T are taken into account in the first multiset comparison.

Definition 8.12 *Given a relation $\sqsupset$ on terms and a function symbol f, we define the relation $\sqsupset^f$ on multisets of terms as follows:*

$$S \sqsupset^f T \overset{\text{def}}{\iff} S{\restriction}_f^{\not<} \sqsupset^{\mathsf{mul}} T{\restriction}_f^{\not<} \uplus T{\restriction}_{\mathcal{V}} - S{\restriction}_{\mathcal{V}}$$

Note that $\sqsupset^f$ depends on a precedence $>$. Whenever we use $\sqsupset^f$, $>$ is defined.

Definition 8.13 *Let $>$ be a precedence and $\langle w, w_0\rangle$ a weight function. First we extend the orders $=_{\mathsf{kv}}$ and $>_{\mathsf{kv}}$ as follows:*

- *$s =_{\mathsf{kv}} t$ if $|s|_x = |t|_x$ for all $x \in \mathcal{V}$, $w(s) = w(t)$, and $\mathsf{root}(s) = \mathsf{root}(t)$,*

- *$s >_{\mathsf{kv}} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either $w(s) > w(t)$ or both $w(s) = w(t)$ and $\mathsf{root}(s) > \mathsf{root}(t)$.*

The order $>_{\mathsf{KV}}$ is now inductively defined as follows: $s >_{\mathsf{KV}} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either $w(s) > w(t)$, or $w(s) = w(t)$ and one of the following alternatives holds:

0. *$s = f^k(t)$ and $t \in \mathcal{V}$ for some $k > 0$,*

1. *$s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$, and $f > g$,*

2. *$s = f(s_1, \dots, s_n)$, $t = f(t_1, \dots, t_n)$, $f \notin \mathcal{F}_{\mathsf{AC}}$, $[s_1, \dots, s_n] >^{\mathsf{lex}}_{\mathsf{KV}} [t_1, \dots, t_n]$,*

3. *$s = f(s_1, s_2)$, $t = f(t_1, t_2)$, $f \in \mathcal{F}_{\mathsf{AC}}$, and for $S = \nabla_f(s)$ and $T = \nabla_f(t)$*
   - *(a) $S >^f_{\mathsf{kv}} T$, or*
   - *(b) $S =^f_{\mathsf{kv}} T$ and $|S| > |T|$, or*
   - *(c) $S =^f_{\mathsf{kv}} T$, $|S| = |T|$, and $S >^{\mathsf{mul}}_{\mathsf{KV}} T$.*

Here $\approx_{\mathsf{AC}}$ is used as preorder in $>^{\mathsf{lex}}_{\mathsf{KV}}$ and $>^{\mathsf{mul}}_{\mathsf{KV}}$ whereas $=_{\mathsf{kv}}$ is used in $>^{\mathsf{mul}}_{\mathsf{kv}}$.

Contrary to what is claimed in [46], the order $>_{\mathsf{KV}}$ of Definition 8.13 is not a simplification order because it lacks the monotonicity property (i.e., $>_{\mathsf{KV}}$ is not closed under contexts), as shown in the following example.

Example 8.14 *Let $\circ$ be an AC symbol and f a unary function symbol with $w(\mathsf{f}) = 0$ and $\mathsf{f} > \circ$. We obviously have*

$$\mathsf{f}(x) >_{\mathsf{KV}} x$$

However, the following does not hold:

$$\mathsf{f}(x) \circ y >_{\mathsf{KV}} x \circ y$$

Let $S = \nabla_\circ(s) = \{\mathsf{f}(x), y\}$ and $T = \nabla_\circ(t) = \{x, y\}$. We have $S{\restriction}^{\not<}_\circ = \{\mathsf{f}(x)\}$, $S{\restriction}_{\mathcal{V}} = \{y\}$, $T{\restriction}^{\not<}_\circ = \varnothing$, and $T{\restriction}_{\mathcal{V}} = \{x, y\}$. Note that $\mathsf{f}(x) >_{\mathsf{kv}} x$ does not hold since $\mathsf{f} \not> x$. Hence case (3a) in Definition 8.13 does not apply. However, cases (3b) and (3c) also do not apply, since $\mathsf{f}(x) =_{\mathsf{kv}} x$ does not hold.

This example does not refute the soundness of $>_{\mathsf{KV}}$ for proving AC termination; note that also $\mathsf{f}(x, y) >_{\mathsf{KV}} \mathsf{f}(\mathsf{g}(x), y)$ does not hold. We prove soundness by extending $>_{\mathsf{KV}}$ to $>_{\mathsf{KV}'}$ which has all desired properties.

Definition 8.15 *The order $>_{\mathsf{KV}'}$ is obtained as in Definition 8.13 after replacing $=^f_{\mathsf{kv}}$ by $\geq^f_{\mathsf{kv}'}$ in cases (3b) and (3c), and using $\geq_{\mathsf{kv}'}$ as preorder in $>^{\mathsf{mul}}_{\mathsf{kv}}$ in case (3a). Here the relation $\geq_{\mathsf{kv}'}$ is defined as follows:*

- *$s \geq_{\mathsf{kv}'} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either $w(s) > w(t)$, or $w(s) = w(t)$ and either $\mathsf{root}(s) \geq \mathsf{root}(t)$ or $t \in \mathcal{V}$.*

Note that $\geq_{\mathsf{kv}'}$ is a preorder that contains $\approx_{\mathsf{AC}}$.

Example 8.16 *Consider again Example 8.14. We have*

$$\mathsf{f}(\mathsf{g}(x), y) >_{\mathsf{KV}'} \mathsf{f}(x, y)$$

because now case (3c) applies: $S\restriction^{\not\approx}_{\mathsf{f}} = \{\mathsf{g}(x)\} \geq^{\mathsf{mul}}_{\mathsf{kv}'} \{x\} = T\restriction^{\not\approx}_{\mathsf{f}} \uplus T\restriction_{\mathcal{V}} - S\restriction_{\mathcal{V}}$, $|S| = 2 = |T|$, and $S = \{\mathsf{g}(x), y\} >^{\mathsf{mul}}_{\mathsf{KV}'} \{x, y\} = T$ because $\mathsf{g}(x) >_{\mathsf{KV}'} x$.

Note that the use of the preorder $\geq_{\mathsf{kv}'}$ is essential for the non-ground version of $>_{\mathsf{KV}}$ to be closed under contexts, even if there is no unary symbol of weight 0. This is illustrated by the following example.

Example 8.17 *Let $\circ$ be an AC symbol, c a constant, f a unary function symbol and g a non-AC binary symbol such that $w(\mathsf{c}) = w_0$, $w(\mathsf{f}) > 0$, and $\mathsf{g} > \circ > \mathsf{c}$. By case 2, we have*

$$\ell = \mathsf{g}(\mathsf{f}(\mathsf{c}), x) >_{\mathsf{KV}} \mathsf{g}(\mathsf{c}, \mathsf{f}(\mathsf{c})) = r$$

However, the following does not hold:

$$s = \ell \circ \mathsf{c} >_{\mathsf{KV}} r \circ \mathsf{c} = t$$

Let $S = \nabla_\circ(s) = \{\ell, \mathsf{c}\}$ and $T = \nabla_\circ(t) = \{r, \mathsf{c}\}$. We have $S\restriction^{\not\approx}_{\circ} = \{\ell\}$, $T\restriction^{\not\approx}_{\circ} = \{r\}$, and $S\restriction_{\mathcal{V}} = T\restriction_{\mathcal{V}} = \varnothing$. Note that $\ell >_{\mathsf{kv}} r$ does not hold since $w(\ell) = w(r)$ and $\mathsf{root}(\ell) = \mathsf{g} = \mathsf{root}(r)$. Hence case (3a) in Definition 8.13 does not apply. On the other hand, $\ell =_{\mathsf{kv}} r$ does not hold since $|\ell|_x = 1 \neq 0 = |r|_x$, excluding cases (3b) and (3c).

In Section 8.5.2, we prove that $>_{\mathsf{KV}'}$ is an AC-compatible simplification order.

Theorem 8.18 *The order $>_{\mathsf{KV}'}$ is an AC-compatible simplification order.*

Since the inclusion $>_{\mathsf{KV}} \subseteq >_{\mathsf{KV}'}$ obviously holds, it follows that $>_{\mathsf{KV}}$ is a sound method for establishing AC termination, despite the lack of monotonicity.

Unfortunately, the order $>_{\mathsf{KV}'}$ lacks one important feature: a polynomial-time algorithm to decide $s >_{\mathsf{KV}'} t$ when the precedence and weight function are given. In Section 8.6.1 we show that the problem is NP-hard. Note that for KBO the problem is known to be linear [57].

8.4 AC-KBO

In this section we present another AC-compatible simplification order. In contrast to $>_{\mathsf{KV}'}$, our new order $>_{\mathsf{ACKBO}}$ contains $>_{\mathsf{S}}$. Moreover, its definition is simpler than $>_{\mathsf{KV}'}$ since we avoid the use of an auxiliary order in case (3b). Finally, $>_{\mathsf{ACKBO}}$ is decidable in polynomial-time. Hence it will be used as the basis for the extension discussed in Section 8.7.

Definition 8.19 *Let $>$ be a precedence and $\langle w, w_0 \rangle$ a weight function. We define $>_{\mathsf{ACKBO}}$ inductively as follows: $s >_{\mathsf{ACKBO}} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either $w(s) > w(t)$, or $w(s) = w(t)$ and one of the following alternatives holds:*

0. $s = f^k(t)$ and $t \in \mathcal{V}$ for some $k > 0$.

1. $s = f(s_1, \dots, s_n)$, $t = g(t_1, \dots, t_m)$, and $f > g$.

2. $s = f(s_1, \dots, s_n)$, $t = f(t_1, \dots, t_n)$, $f \notin \mathcal{F}_{\mathsf{AC}}$, $[s_1, \dots, s_n] >^{\mathsf{lex}}_{\mathsf{ACKBO}} [t_1, \dots, t_n]$.

3. $s = f(s_1, s_2)$, $t = f(t_1, t_2)$, $f \in \mathcal{F}_{\mathsf{AC}}$, and for $S = \triangledown_f(s)$ and $T = \triangledown_f(t)$,

  (a) $S >^{f}_{\mathsf{ACKBO}} T$, or

  (b) $S \approx^{f}_{\mathsf{AC}} T$, and $|S| > |T|$, or

  (c) $S \approx^{f}_{\mathsf{AC}} T$, $|S| = |T|$, and $S{\restriction}^{<}_{f} >^{\mathsf{mul}}_{\mathsf{ACKBO}} T{\restriction}^{<}_{f}$.

The relation $\approx_{\mathsf{AC}}$ is used as preorder in $>^{\mathsf{lex}}_{\mathsf{ACKBO}}$ and $>^{\mathsf{mul}}_{\mathsf{ACKBO}}$.

Note that in case (3c) we compare the multisets $S{\restriction}^{<}_{f}$ and $T{\restriction}^{<}_{f}$ rather than S and T in the multiset extension of $>_{\mathsf{ACKBO}}$. In Section 8.5.1, we prove the following result stating correctness of our order:

Theorem 8.20 *The order $>_{\mathsf{ACKBO}}$ is an AC-compatible simplification order.*

Since we deal with finite non-variadic signatures, simplification orders are well-founded. The following theorem is easily proved.

Theorem 8.21 *If $>$ is total then $>_{\mathsf{ACKBO}}$ is AC-total on ground terms.* □

The following example shows that AC-KBO is not *incremental*, i.e., orientability is not necessarily preserved when the precedence is extended. This is in contrast to the AC-RPO of Rubio [65]. However, this is not necessarily a disadvantage; actually, the example shows that by allowing partial precedences more TRSs can be proved to be AC terminating using AC-KBO.

Example 8.22 *Consider the TRS $\mathcal{R}$ consisting of the rules*

$$\mathsf{a} \circ (\mathsf{b} \bullet \mathsf{c}) \to \mathsf{b} \circ \mathsf{f}(\mathsf{a} \bullet \mathsf{c}) \qquad\qquad \mathsf{a} \bullet (\mathsf{b} \circ \mathsf{c}) \to \mathsf{b} \bullet \mathsf{f}(\mathsf{a} \circ \mathsf{c})$$

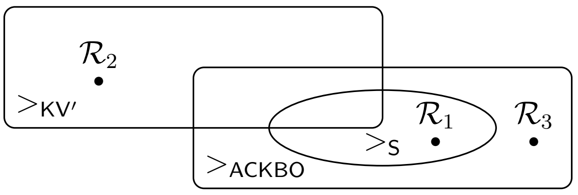
over the signature $\mathcal{F} = \{\mathsf{a}, \mathsf{b}, \mathsf{c}, \mathsf{f}, \circ, \bullet\}$ with $\circ, \bullet \in \mathcal{F}_{\mathsf{AC}}$. By taking the precedence $\mathsf{f} > \mathsf{a}, \mathsf{b}, \mathsf{c}, \circ, \bullet$ and admissible weight function $\langle w, w_0 \rangle$ with $w(\mathsf{f}) = w(\circ) = w(\bullet) = 0$, $w_0 = w(\mathsf{a}) = w(\mathsf{c}) = 1$, and $w(\mathsf{b}) = 2$, the resulting $>_{\mathsf{ACKBO}}$ orients both rules from left to right. It is essential that $\circ$ and $\bullet$ are incomparable in the precedence: We must have $w(\mathsf{f}) = 0$, so $\mathsf{f} > \mathsf{a}, \mathsf{b}, \mathsf{c}, \circ, \bullet$ is enforced by admissibility. If $\circ > \bullet$ then the first rule can only be oriented from left to right if $\mathsf{a} >_{\mathsf{ACKBO}} \mathsf{f}(\mathsf{a} \bullet \mathsf{c})$ holds, which contradicts the subterm property. If $\bullet > \circ$ then we use the second rule to obtain the impossible $\mathsf{a} >_{\mathsf{ACKBO}} \mathsf{f}(\mathsf{a} \bullet \mathsf{c})$. Similarly, $\mathcal{R}$ is also orientable by $>_{\mathsf{KV'}}$ but we must adopt a non-total precedence.

In the rest of this section, we investigate relationships between our AC-KBO and other variants. In particular, Steinbach's order is a special case of our AC-KBO. As a consequence, correctness of $>_{\mathsf{S}}$ (i.e., Theorem 8.7) is concluded by Theorem 8.20.

Theorem 8.23 *If every AC symbol has minimal precedence then $>_{\mathsf{S}} = >_{\mathsf{ACKBO}}$.*

Proof. Suppose that every function symbol in $\mathcal{F}_{\mathsf{AC}}$ is minimal with respect to $>$. We show that $s >_{\mathsf{S}} t$ iff $s >_{\mathsf{ACKBO}} t$ by induction on s. It is clearly sufficient to consider case 3 in Definition 8.5 and cases (3a)–(3c) in Definition 8.19. So let $s = f(s_1, s_2)$ and $t = f(t_1, t_2)$ such that $w(s) = w(t)$ and $f \in \mathcal{F}_{\mathsf{AC}}$. Let $S = \nabla_f(s)$ and $T = \nabla_f(t)$.

- Let $s >_{\mathsf{S}} t$ by case 3. We have $S >_{\mathsf{S}}^{\mathsf{mul}} T$. Since $S >_{\mathsf{S}}^{\mathsf{mul}} T$ involves only comparisons $s' >_{\mathsf{S}} t'$ for subterms s' of s, the induction hypothesis yields $S >_{\mathsf{ACKBO}}^{\mathsf{mul}} T$. Because f is minimal in $>$, $S = S{\restriction}_f^{\not<} \uplus S{\restriction}_{\mathcal{V}}$ and $T = T{\restriction}_f^{\not<} \uplus T{\restriction}_{\mathcal{V}}$. For no elements $u \in S{\restriction}_{\mathcal{V}}$ and $v \in T{\restriction}_f^{\not<}$, $u >_{\mathsf{ACKBO}} v$ or $u \approx_{\mathsf{AC}} v$ holds. Hence $S >_{\mathsf{ACKBO}}^{\mathsf{mul}} T$ implies $S >_{\mathsf{ACKBO}}^{f} T$ or both $S \approx_{\mathsf{AC}}^{f} T$ and $S{\restriction}_{\mathcal{V}} \supsetneq T{\restriction}_{\mathcal{V}}$. In the former case $s >_{\mathsf{ACKBO}} t$ is due to case (3a) in Definition 8.19. In the latter case we have $|S| > |T|$ and $s >_{\mathsf{ACKBO}} t$ follows by case (3b).
- Let $s >_{\mathsf{ACKBO}} t$ by applying one of the cases (3a)–(3c) in Definition 8.19.
  - Suppose (3a) applies. Then we have $S >_{\mathsf{ACKBO}}^{f} T$. Since f is minimal in $>$, $S{\restriction}_f^{\not<} = S - S{\restriction}_{\mathcal{V}}$ and $T{\restriction}_f^{\not<} \uplus T{\restriction}_{\mathcal{V}} = T$. Hence $S >_{\mathsf{ACKBO}}^{\mathsf{mul}} (T - S{\restriction}_{\mathcal{V}}) \uplus S{\restriction}_{\mathcal{V}} \supseteq T$. We obtain $S >_{\mathsf{S}}^{\mathsf{mul}} T$ from the induction hypothesis and thus case 3 in Definition 8.5 applies.



$\mathcal{R}_1$	Example 8.6 (and 8.11)
$\mathcal{R}_2$	Example 8.11
$\mathcal{R}_3$	Example 8.24

Figure 8.1: Comparison.

– Suppose (3b) applies. Analogous to the previous case, the inclusion $S \approx_{\mathsf{AC}}^{\mathsf{mul}} (T - S{\upharpoonright_{\mathcal{V}}}) \uplus S{\upharpoonright_{\mathcal{V}}} \supseteq T$ holds. Since $|S| > |T|$, $S \approx_{\mathsf{AC}}^{\mathsf{mul}} T$ is not possible. Thus $(T - S{\upharpoonright_{\mathcal{V}}}) \uplus S{\upharpoonright_{\mathcal{V}}} \supsetneq T$ and hence $S >_{\mathsf{S}}^{\mathsf{mul}} T$.

– If case (3c) applies then $S{\upharpoonright_f^{<}} >_{\mathsf{ACKBO}}^{\mathsf{mul}} T{\upharpoonright_f^{<}}$. This is impossible since both sides are empty as f is minimal in $>$. □

The following example shows that $>_{\mathsf{ACKBO}}$ is a proper extension of $>_{\mathsf{S}}$ and incomparable with $>_{\mathsf{KV}'}$.

Example 8.24 *Consider the TRS $\mathcal{R}_3$ consisting of the rules*

$$\begin{array}{cc}
\mathsf{f}(x+y) \to \mathsf{f}(x)+y & \mathsf{g}(x)+y \to \mathsf{g}(x+y) \\
\mathsf{h}(\mathsf{a},\mathsf{b}) \to \mathsf{h}(\mathsf{b},\mathsf{a}) & \mathsf{h}(\mathsf{g}(\mathsf{a}),\mathsf{a}) \to \mathsf{h}(\mathsf{a},\mathsf{g}(\mathsf{b})) \\
\mathsf{h}(\mathsf{g}(\mathsf{a}),\mathsf{b}) \to \mathsf{h}(\mathsf{a},\mathsf{g}(\mathsf{a})) & \mathsf{h}(\mathsf{a},\mathsf{g}(\mathsf{g}(\mathsf{a}))) \to \mathsf{h}(\mathsf{g}(\mathsf{a}),\mathsf{f}(\mathsf{a})) \\
\mathsf{f}(\mathsf{a})+\mathsf{g}(\mathsf{b}) \to \mathsf{f}(\mathsf{b})+\mathsf{g}(\mathsf{a}) &
\end{array}$$

over the signature $\{+,\mathsf{f},\mathsf{g},\mathsf{h},\mathsf{a},\mathsf{b}\}$ with $+ \in \mathcal{F}_{\mathsf{AC}}$. Consider the precedence $\mathsf{f} > + > \mathsf{g} > \mathsf{a} > \mathsf{b} > \mathsf{h}$ together with the admissible weight function $\langle w, w_0\rangle$ with $w(+) = w(\mathsf{h}) = 0$, $w(\mathsf{f}) = w(\mathsf{a}) = w(\mathsf{b}) = w_0 = 1$ and $w(\mathsf{g}) = 2$. The interesting rule is $\mathsf{f}(\mathsf{a})+\mathsf{g}(\mathsf{b}) \to \mathsf{f}(\mathsf{b})+\mathsf{g}(\mathsf{a})$. For $S = \nabla_+(\mathsf{f}(\mathsf{a})+\mathsf{g}(\mathsf{b}))$ and $T = \nabla_+(\mathsf{f}(\mathsf{b})+\mathsf{g}(\mathsf{a}))$ the multisets $S' = S{\upharpoonright_+^{\not<}} = \{\mathsf{f}(\mathsf{a})\}$ and $T' = T{\upharpoonright_+^{\not<}} \uplus T{\upharpoonright_{\mathcal{V}}} - S{\upharpoonright_{\mathcal{V}}} = \{\mathsf{f}(\mathsf{b})\}$ satisfy $S' >_{\mathsf{ACKBO}}^{\mathsf{mul}} T'$ as $\mathsf{f}(\mathsf{a}) >_{\mathsf{ACKBO}} \mathsf{f}(\mathsf{b})$, so that case (3a) of Definition 8.19 applies. All other rules are oriented from left to right by both $>_{\mathsf{KV}'}$ and $>_{\mathsf{ACKBO}}$, and they enforce a precedence and weight function which are identical (or very similar) to the one given above. Since $>_{\mathsf{KV}'}$ orients the rule $\mathsf{f}(\mathsf{a})+\mathsf{g}(\mathsf{b}) \to \mathsf{f}(\mathsf{b})+\mathsf{g}(\mathsf{a})$ from right to left, $\mathcal{R}_3$ cannot be compatible with $>_{\mathsf{KV}'}$. It is easy to see that the rule $\mathsf{g}(x)+y \to \mathsf{g}(x+y)$ requires $+ > \mathsf{g}$, and hence $>_{\mathsf{S}}$ cannot be applied.

We close this section with Figure 8.1 that summarizes the relationships between the orders introduced so far.

8.5 Correctness

In this section, we show that $>_{\mathsf{ACKBO}}$ and $>_{\mathsf{KV}'}$ are AC-compatible simplification orders.

8.5.1 Correctness of AC-KBO

In order to prove correctness of $>_{\mathsf{ACKBO}}$, we first show that $\langle \approx_{\mathsf{AC}}, >_{\mathsf{ACKBO}} \rangle$ is an order pair. To facilitate the proof, we decompose $>_{\mathsf{ACKBO}}$ into several orders. We write

- $s >_{01} t$ if $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$ and either $w(s) > w(t)$ or $w(s) = w(t)$ and case 0 or case 1 of Definition 8.19 applies,
- $s >_{23,k} t$ if $|s|, |t| \leq k$, $|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$, $w(s) = w(t)$, and case 2 or case 3 applies.

The union of $>_{01}$ and $>_{23,k}$ is denoted by $>_k$. The next lemma states straightforward properties.

Lemma 8.25 *The following statements hold:*

1. $>_{\mathsf{ACKBO}} \; = \bigcup \{ >_k \mid k \in \mathbb{N} \}$,
2. $\langle \approx_{\mathsf{AC}}, >_{01} \rangle$ *is an order pair, and*
3. $(>_{01} \cdot >_k) \cup (>_k \cdot >_{01}) \; \subseteq \; >_{01}$.

Proof.

1. The inclusion from right to left is obvious from the definition. For the inclusion from left to right, suppose $s >_{\mathsf{ACKBO}} t$. If either $w(s) > w(t)$, or $w(s) = w(t)$ and case 0 or case 1 of Definition 8.19 applies, then trivially $s >_{01} t$. If case 2 or case 3 applies, then $s >_{23,k} t$ for any k with $k \geq \max(|s|, |t|)$.

2. First we show that $>_{01}$ is transitive. Suppose $s >_{01} t >_{01} u$. If $w(s) > w(t)$ or $w(t) > w(u)$, then $w(s) > w(u)$ and $s >_{01} u$. Hence suppose $w(s) = w(t) = w(u)$. Since $s, t \notin \mathcal{V}$, we may write $s = f(s_1, \ldots, s_n)$ and $t = g(t_1, \ldots, t_m)$ with $f > g$. Because of admissibility, g is not a unary symbol with $w(g) = 0$. Thus $u \notin \mathcal{V}$, and we may write $u = h(u_1, \ldots, u_l)$

with $g > h$. By the transitivity of $>$ we obtain $s >_{01} u$. The irreflexivity of $>_{01}$ is obvious from the definition. It remains to show the compatibility condition $\approx_{\mathsf{AC}} \cdot >_{01} \cdot \approx_{\mathsf{AC}} \subseteq >_{01}$. This easily follows from the fact that $w(s) = w(t)$ and $\mathsf{root}(s) = \mathsf{root}(t)$ whenever $s \approx_{\mathsf{AC}} t$.

3. Suppose $s = f(s_1, \ldots, s_n) >_{01} t = g(t_1, \ldots, t_m) >_k u$. If $t >_{01} u$ then $s >_{01} u$ follows from the transitivity of $>_{01}$. Suppose $t >_{23,k} u$. So $w(t) = w(u)$. Thus $w(s) > w(u)$ if $w(s) > w(t)$, and case 1 applies if $w(s) = w(t)$. The inclusion $>_k \cdot >_{01} \subseteq >_k$ is proved in exactly the same way. □

Lemma 8.26 *Let $>$ be a precedence, $f \in \mathcal{F}$, and $\langle \succsim, \succ \rangle$ an order pair on terms. Then $\langle \succsim^f, \succ^f \rangle$ is an order pair on multisets of terms.*

Proof. We first prove compatibility. Suppose $S \succsim^f T \succ^f U$. From $T \succ^f U$ we infer that $T{\restriction}_f^{\not<} \uplus T{\restriction}_{\mathcal{V}} \succ^{\mathsf{mul}} U{\restriction}_f^{\not<} \uplus U{\restriction}_{\mathcal{V}}$. Hence $S{\restriction}_f^{\not<} \succ^{\mathsf{mul}} U{\restriction}_f^{\not<} \uplus U{\restriction}_{\mathcal{V}} - S{\restriction}_{\mathcal{V}}$ follows from $S \succsim^f T$. Hence also $S\ (\succsim \cdot \succ)^f\ U$. We obtain the desired $S \succ^f U$ from the compatibility of $\succsim$ and $\succ$. Transitivity of $\succsim^f$ and $\succ^f$ is obtained in a very similar way. Reflexivity of $\succsim^f$ and irreflexivity of $\succ^f$ are obvious. □

We employ the following simple criterion to construct order pairs, which enables us to prove correctness in a modular way.

Lemma 8.27 *Let $\langle \succsim, \succ_k \rangle$ be order pairs for $k \in \mathbb{N}$ with $\succ_k \subseteq \succ_{k+1}$. If $\succ$ is the union of all $\succ_k$, then $\langle \succsim, \succ \rangle$ is an order pair.*

Proof. The relation $\succsim$ is a preorder by assumption. Suppose $s \succ t \succ u$. By assumption there exist k and l such that $s \succ_k t \succ_l u$. Let $m = \max(k, l)$. We obtain $s \succ_m t \succ_m u$ from the assumptions of the lemma and hence $s \succ_m u$ follows from the fact that $\langle \succsim, \succ_m \rangle$ is an order pair. Compatibility is an immediate consequence of the assumptions and the irreflexivity of $\succ$ is obtained by an easy induction proof. □

Lemma 8.28 *The pair $\langle \approx_{\mathsf{AC}}, >_{\mathsf{ACKBO}} \rangle$ is an order pair.*

Proof. According to Lemmata 8.27 and 8.25(1), it is sufficient to prove that $\langle \approx_{\mathsf{AC}}, >_k \rangle$ is an order pair for all $k \in \mathbb{N}$. Due to Lemma 8.25(2,3) it suffices to prove that $\langle \approx_{\mathsf{AC}}, >_{23,k} \rangle$ is an order pair, which follows by using induction on k in combination with Lemma 8.26 and Theorem 2.7. □

The subterm property is an easy consequence of transitivity and admissibility.

Lemma 8.29 *The order $>_{\mathsf{ACKBO}}$ has the subterm property.* $\square$

Next we prove that $>_{\mathsf{ACKBO}}$ is closed under contexts. The following lemma is an auxiliary result needed for its proof. In order to reuse this lemma for the correctness proof of $>_{\mathsf{KV'}}$ in Section 8.5.2, we prove it in an abstract setting.

Lemma 8.30 *Let $\langle \succsim, \succ \rangle$ be an order pair on terms and $f \in \mathcal{F}_{\mathsf{AC}}$ with $f(u,v) \succ u, v$ for arbitrary terms u and v. If $s \succsim t$ then $\{s\} \succsim^{\mathsf{mul}} \nabla_f(t)$ or $\{s\} \succ^{\mathsf{mul}} \nabla_f(t)$. If $s \succ t$ then $\{s\} \succ^{\mathsf{mul}} \nabla_f(t)$.*

Proof. Let $\nabla_f(t) = \{t_1, \dots, t_m\}$. If $m = 1$ then $\nabla_f(t) = \{t\}$ and the lemma holds trivially. Otherwise we get $t \succ t_j$ for every $j \in \{1, \dots, m\}$ by recursively applying the assumption. Hence $s \succ t_j$ by the transitivity of $\succ$ or the compatibility of $\succ$ and $\succsim$. We conclude that $\{s\} \succ^{\mathsf{mul}} \nabla_f(t)$. $\square$

In the following proof of closure under contexts, admissibility is essential. This is in contrast to the corresponding result for standard KBO.

Lemma 8.31 *If $\langle w, w_0 \rangle$ is admissible for $>$ then $>_{\mathsf{ACKBO}}$ is closed under contexts.*

Proof. Suppose $s >_{\mathsf{ACKBO}} t$. We consider the context $h(\square, u)$ with $h \in \mathcal{F}_{\mathsf{AC}}$ and u an arbitrary term, and prove that $s' = h(s,u) >_{\mathsf{ACKBO}} h(t,u) = t'$. Closure under contexts of $>_{\mathsf{ACKBO}}$ follows then by induction; contexts rooted by a non-AC symbol are handled as in the proof for standard KBO.

If $w(s) > w(t)$ then obviously $w(s') > w(t')$. So we assume $w(s) = w(t)$. Let $S = \nabla_h(s)$, $T = \nabla_h(t)$, and $U = \nabla_h(u)$. Note that $\nabla_h(s') = S \uplus U$ and $\nabla_h(t') = T \uplus U$. Because $>^{\mathsf{mul}}_{\mathsf{ACKBO}}$ is closed under multiset sum, it suffices to show that one of the cases (3a)–(3c) of Definition 8.19 holds for S and T. Let $f = \mathsf{root}(s)$ and $g = \mathsf{root}(t)$. We distinguish the following cases.

- Suppose $f \not< h$. We have $S = S{\restriction}_h^{\not<} = \{s\}$, and from Lemmata 8.29 and 8.30 we obtain $S >^{\mathsf{mul}}_{\mathsf{ACKBO}} T$. Since T is a superset of $T{\restriction}_h^{\not<} \uplus T{\restriction}_{\mathcal{V}} - S{\restriction}_{\mathcal{V}}$, (3a) applies.

- Suppose $f = h > g$. We have $T{\restriction}_h^{\not<} \uplus T{\restriction}_{\mathcal{V}} = \varnothing$. If $S{\restriction}_h^{\not<} \neq \varnothing$, then (3a) applies. Otherwise, since AC symbols are binary and $T = \{t\}$, $|S| \geq 2 > 1 = |T|$. Hence (3b) applies.

- Suppose $f = g = h$. Then, $s >_{\mathsf{ACKBO}} t$ must be derived by one of the cases (3a)–(3c) for S and T.
- Suppose $f, g < h$. We have $S{\restriction}^{\not<}_{h} = T{\restriction}^{\not<}_{h} \uplus T{\restriction}_{\mathcal{V}} = \varnothing$, $|S| = |T| = 1$, and $S{\restriction}^{<}_{h} = \{s\} >^{\mathsf{mul}}_{\mathsf{ACKBO}} \{t\} = T{\restriction}^{<}_{h}$. Hence (3c) holds.

Note that $f \geq g$ since $w(s) = w(t)$ and $s >_{\mathsf{ACKBO}} t$. Moreover, if $t \in \mathcal{V}$ then $s = f^k(t)$ for some $k > 0$ with $w(f) = 0$, which entails $f > h$ due to admissibility. □

Closure under substitutions is the trickiest part since by substituting AC-rooted terms for variables that appear in the top-flattening of a term, the structure of the term changes. In the proof, the multisets $\{t \in T \mid t \notin \mathcal{V}\}$, $\{t\sigma \mid t \in T\}$, and $\{\nabla_f(t) \mid t \in T\}$ are denoted by $T{\restriction}_{\mathcal{F}}$, $T\sigma$, and $\nabla_f(T)$, respectively.

Lemma 8.32 *Let $>$ be a precedence, $f \in \mathcal{F}_{\mathsf{AC}}$, and $\langle \succsim, \succ \rangle$ an order pair on terms such that $\succsim$ and $\succ$ are closed under substitutions and $f(x, y) \succ x, y$. Consider terms s and t such that $S = \nabla_f(s)$, $T = \nabla_f(t)$, $S' = \nabla_f(s\sigma)$, and $T' = \nabla_f(t\sigma)$.*

1. *If $S \succ^f T$ then $S' \succ^f T'$.*
2. *If $S \succsim^f T$ then $S' \succ^f T'$ or $S' \succsim^f T'$. In the latter case $|S| - |T| \leq |S'| - |T'|$ and $S'{\restriction}^{<}_{f} \succ^{\mathsf{mul}} T'{\restriction}^{<}_{f}$ whenever $S{\restriction}^{<}_{f} \succ^{\mathsf{mul}} T{\restriction}^{<}_{f}$.*

Proof. Let v be an arbitrary term. By the assumption on $\succ$ we have either $\{v\} = \nabla_f(v)$ or both $\{v\} \succ^{\mathsf{mul}} \nabla_f(v)$ and $1 < |\nabla_f(v)|$. Hence, for any set V of terms, either $V = \nabla_f(V)$ or both $V \succ^{\mathsf{mul}} \nabla_f(V)$ and $|V| < |\nabla_f(V)|$. Moreover, for $V = \nabla_f(v)$, the following equalities hold:

$$\nabla_f(v\sigma){\restriction}^{\not<}_{f} = V{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(V{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \qquad \nabla_f(v\sigma){\restriction}_{\mathcal{V}} = \nabla_f(V{\restriction}_{\mathcal{V}}\sigma){\restriction}_{\mathcal{V}}$$

To prove the lemma, assume $S \sqsupset^f T$ for $\sqsupset \in \{\succsim, \succ\}$. We have $S{\restriction}^{\not<}_{f} \sqsupset^{\mathsf{mul}} T{\restriction}^{\not<}_{f} \uplus U$ where $U = (T - S){\restriction}_{\mathcal{V}}$. Since multiset extensions preserve closure under substitutions, $S{\restriction}^{\not<}_{f}\sigma \sqsupset^{\mathsf{mul}} T{\restriction}^{\not<}_{f}\sigma \uplus U\sigma$ follows. Using the above (in)equalities, we obtain

$$\begin{aligned}
S'{\restriction}^{\not<}_{f} = {} & S{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(S{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \\
\sqsupset^{\mathsf{mul}} {} & T{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(S{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \uplus U\sigma \\
O \quad & T{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(S{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \uplus \nabla_f(U\sigma) \\
= {} & T{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(S{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \uplus \nabla_f(U\sigma){\restriction}_{\mathcal{V}} \uplus \nabla_f(U\sigma){\restriction}^{\not<}_{f} \uplus \nabla_f(U\sigma){\restriction}^{<}_{f} \\
P \quad & T{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(T{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \uplus \nabla_f(U\sigma){\restriction}_{\mathcal{V}} \\
= {} & T{\restriction}^{\not<}_{f}\sigma \uplus \nabla_f(T{\restriction}_{\mathcal{V}}\sigma){\restriction}^{\not<}_{f} \uplus \nabla_f(T{\restriction}_{\mathcal{V}}\sigma){\restriction}_{\mathcal{V}} - \nabla_f(S{\restriction}_{\mathcal{V}}\sigma){\restriction}_{\mathcal{V}} \\
= {} & T'{\restriction}^{\not<}_{f} \uplus T'{\restriction}_{\mathcal{V}} - S'{\restriction}_{\mathcal{V}}
\end{aligned}$$

Here O denotes $=$ if $U\sigma = \nabla_f(U\sigma)$ and $\succ^{\mathsf{mul}}$ if $|U\sigma| < |\nabla_f(U\sigma)|$, while P denotes $=$ if $U\sigma{\restriction}_f^{<} = \varnothing$ and $\supsetneq$ otherwise. Since $\langle \succsim^{\mathsf{mul}}, \succ^{\mathsf{mul}} \rangle$ is an order pair with $\supseteq \subseteq \succsim^{\mathsf{mul}}$ and $\supsetneq \subseteq \succ^{\mathsf{mul}}$, we obtain $S' \sqsupset^f T'$.

It remains to show (2). If $S' \not\succ^f T'$ then O and P are both $=$ and thus $U\sigma = \nabla_f(U\sigma)$ and $U\sigma{\restriction}_f^{<} = \varnothing$. Let $X = S{\restriction}_{\mathcal{V}} \cap T{\restriction}_{\mathcal{V}}$. We have $U = T{\restriction}_{\mathcal{V}} - X$.

- Since $|W{\restriction}_{\mathcal{F}}\sigma| = |W{\restriction}_{\mathcal{F}}|$ and $|W| \leq |\nabla_f(W)|$ for an arbitrary set W of terms, we have $|S'| \geq |S| - |X| + |\nabla_f(X\sigma)|$. From $|U\sigma| = |U| = |T{\restriction}_{\mathcal{V}}| - |X|$ we obtain $|T'| = |T{\restriction}_{\mathcal{F}}\sigma| + |\nabla_f(U\sigma)| + |\nabla_f(X\sigma)| = |T| - |X| + |\nabla_f(X\sigma)|$. Hence $|S| - |T| \leq |S'| - |T'|$ as desired.

- Suppose $S{\restriction}_f^{<} \succ^{\mathsf{mul}} T{\restriction}_f^{<}$. From $U\sigma{\restriction}_f^{<} = \varnothing$ we infer $T{\restriction}_{\mathcal{V}}\sigma{\restriction}_f^{<} \subseteq S{\restriction}_{\mathcal{V}}\sigma{\restriction}_f^{<}$. Because $S'{\restriction}_f^{<} = S{\restriction}_f^{<}\sigma \uplus S{\restriction}_{\mathcal{V}}\sigma{\restriction}_f^{<}$ and $T'{\restriction}_f^{<} = T{\restriction}_f^{<}\sigma \uplus T{\restriction}_{\mathcal{V}}\sigma{\restriction}_f^{<}$, closure under substitutions of $\succ^{\mathsf{mul}}$ (which it inherits from $\succ$ and $\succsim$) yields the desired $S'{\restriction}_f^{<} \succ^{\mathsf{mul}} T'{\restriction}_f^{<}$. □

Lemma 8.33 $>_{\mathsf{ACKBO}}$ *is closed under substitutions.*

Proof. If $s >_{\mathsf{ACKBO}} t$ is obtained by cases 0–1 in Definition 8.19, the proof for standard KBO goes through. If (3a) or (3b) is used to obtain $s >_{\mathsf{ACKBO}} t$, according to Lemma 8.32 one of these cases also applies to $s\sigma >_{\mathsf{ACKBO}} t\sigma$. The final case is (3c). So $\nabla_f(s){\restriction}_f^{<} >_{\mathsf{ACKBO}}^{\mathsf{mul}} \nabla_f(t){\restriction}_f^{<}$. Suppose $s\sigma >_{\mathsf{ACKBO}} t\sigma$ cannot be obtained by (3a) or (3b). Lemma 8.32(2) yields $|\nabla_f(s\sigma)| = |\nabla_f(t\sigma)|$ and $\nabla_f(s\sigma){\restriction}_f^{<} >_{\mathsf{ACKBO}}^{\mathsf{mul}} \nabla_f(t\sigma){\restriction}_f^{<}$. Hence, case (3c) is applicable and we obtain $s\sigma >_{\mathsf{ACKBO}} t\sigma$. □

8.5.2 Correctness of $>_{\mathsf{KV'}}$

In this section, we prove that $>_{\mathsf{KV'}}$ is an AC-compatible simplification order. The proof mimics the one given above for $>_{\mathsf{ACKBO}}$, but there are some subtle differences. The easy proof of the following lemma is omitted.

Lemma 8.34 *The pairs* $\langle \approx_{\mathsf{AC}}, >_{\mathsf{kv}} \rangle$ *and* $\langle \geq_{\mathsf{kv'}}, >_{\mathsf{kv}} \rangle$ *are order pairs.* □

Lemma 8.35 *The pair* $\langle \approx_{\mathsf{AC}}, >_{\mathsf{KV'}} \rangle$ *is an order pair.*

Proof. Similar to the proof of Lemma 8.28, except for case 3 of Definition 8.15, where we need Lemma 8.34 and Theorem 2.7. □

The subterm property follows exactly as in the proof of Lemma 8.29; note that the relation $>_{01}$ has the subterm property, and we obviously have $>_{01} \subseteq >_{\mathsf{KV}'}$.

Lemma 8.36 *The order $>_{\mathsf{KV}'}$ has the subterm property.* □

Lemma 8.37 *The order $>_{\mathsf{KV}'}$ is closed under contexts.*

Proof. Suppose $s >_{\mathsf{KV}'} t$. We follow the proof for $>_{\mathsf{ACKBO}}$ in Lemma 8.31 and consider here the case that $w(s) = w(t)$. We will show that one of the cases (3a)–(3c) in Definition 8.15 (8.13) is applicable to $S = \nabla_h(s)$ and $T = \nabla_h(t)$. Let $f = \mathsf{root}(s)$ and $g = \mathsf{root}(t)$. The proof proceeds by case splitting according to the derivation of $s >_{\mathsf{KV}'} t$.

- Suppose $s = f^k(t)$ with $k > 0$ and $t \in \mathcal{V}$. By admissibility, f is maximal in the precedence. Hence, $S{\restriction}_h^{\not<} = \{s\} \geq_{\mathsf{kv}'}^{\mathsf{mul}} \{t\}$. We have $|S| = |T| = 1$ and $S >_{\mathsf{KV}'}^{\mathsf{mul}} T$. Hence (3c) applies. (This case breaks down for $>_{\mathsf{KV}}$.)
- Suppose $f = g \notin \mathcal{F}_{\mathsf{AC}}$. We have $S \geq_{\mathsf{kv}'}^{\mathsf{mul}} T$, $|S| = |T| = 1$, and $S = \{s\} >_{\mathsf{KV}'}^{\mathsf{mul}} \{t\} = T$. Hence (3c) applies.
- The remaining cases are similar to the proof of Lemma 8.31, except that we use Lemma 8.30 with $\langle \geq_{\mathsf{kv}'}, >_{\mathsf{kv}} \rangle$. □

For closure under substitutions we need to extend Lemma 8.32 with the following case:

3. *If $S \succsim^f T$ and $S' \not\succ^f T'$ then $S' - T' \supseteq S\sigma - T\sigma$ and $T\sigma - S\sigma \supseteq T' - S'$.*

Proof. We continue the proof of Lemma 8.32. From $\nabla_f(U\sigma) = U\sigma$ we infer that $T' = T{\restriction}_{\mathcal{F}}\sigma \uplus U\sigma \uplus \nabla_f(X\sigma)$. On the other hand, $S' = S{\restriction}_{\mathcal{F}}\sigma \uplus \nabla_f(Y\sigma) \uplus \nabla_f(X\sigma)$ with $Y = S{\restriction}_{\mathcal{V}} - X$. Hence we obtain the following two inclusions:

$$\begin{aligned}
T' - S' &\subseteq T{\restriction}_{\mathcal{F}}\sigma \uplus U\sigma - S{\restriction}_{\mathcal{F}}\sigma \\
&= T{\restriction}_{\mathcal{F}}\sigma \uplus U\sigma \uplus X\sigma - (S{\restriction}_{\mathcal{F}} \uplus X\sigma) \subseteq T\sigma - S\sigma \\
S' - T' &\supseteq S{\restriction}_{\mathcal{F}}\sigma - T{\restriction}_{\mathcal{F}}\sigma - U\sigma \\
&= S{\restriction}_{\mathcal{F}}\sigma \uplus X\sigma - (T{\restriction}_{\mathcal{F}} \uplus U\sigma \uplus X\sigma) \supseteq S\sigma - T\sigma
\end{aligned}$$ □

Lemma 8.38 *The order $>_{\mathsf{KV}'}$ is closed under substitutions.*

Proof. By induction on $|s|$, we verify that $s >_{\mathsf{KV}'} t$ implies $s\sigma >_{\mathsf{KV}'} t\sigma$. If $s >_{\mathsf{KV}'} t$ is derived by one of the cases 0, 1, 2, (3a), or (3b) in Definition 8.15 (8.13), then the proof of Lemma 8.31 goes through. So suppose that $s >_{\mathsf{KV}'} t$ is derived by case (3c) and further suppose that $s\sigma >_{\mathsf{KV}'} t\sigma$ can be derived neither by case (3a) nor (3b). By definition we have $\nabla_f(s) >^{\mathsf{mul}}_{\mathsf{KV}'} \nabla_f(t)$. This is equivalent[6] to

$$\nabla_f(s) - \nabla_f(t) >^{\mathsf{mul}}_{\mathsf{KV}'} \nabla_f(t) - \nabla_f(s)$$

We obtain $\nabla_f(s)\sigma - \nabla_f(t)\sigma >^{\mathsf{mul}}_{\mathsf{KV}'} \nabla_f(t)\sigma - \nabla_f(s)\sigma$ from the induction hypothesis and thus $\nabla_f(s\sigma) - \nabla_f(t\sigma) >^{\mathsf{mul}}_{\mathsf{KV}'} \nabla_f(t\sigma) - \nabla_f(s\sigma)$ by Lemma 8.32(3). Using the earlier equivalence, we infer $\nabla_f(s\sigma) >^{\mathsf{mul}}_{\mathsf{KV}'} \nabla_f(t\sigma)$ and hence case (3c) applies to obtain the desired $s\sigma >_{\mathsf{KV}'} t\sigma$. □

The combination of the above results proves Theorem 8.18.

8.6 Complexity

In this section, we investigate complexities of membership problems for the AC-compatible variants of KBO.

8.6.1 Membership Problem for $>_{\mathsf{KV}'}$

Deciding $s >_{\mathsf{KV}'} t$ is considerably difficult in theory; we show NP-hardness of this problem by adapting the reduction technique of [76, Theorem 4.2].

Theorem 8.39 *The decision problem for $>_{\mathsf{KV}'}$ is NP-hard.*

Proof. It is sufficient to show NP-hardness of deciding $S >^{\mathsf{mul}}_{\mathsf{kv}'} T$ since we can easily construct terms s and t such that $S >^{\mathsf{mul}}_{\mathsf{kv}'} T$ iff $s >_{\mathsf{KV}'} t$. To wit, for $S = \{s_1, \ldots, s_n\}$ and $T = \{t_1, \ldots, t_m\}$ we introduce an AC symbol $\circ$ and constants c and d such that $\circ > \mathsf{c}, \mathsf{d}$ and define

$$s = s_1 \circ \cdots \circ s_n \circ \mathsf{c} \qquad\qquad t = t_1 \circ \cdots \circ t_m \circ \mathsf{d} \circ \mathsf{d}$$

The weights of c and d should be chosen so that $w(s) = w(t)$. If $S >^{\mathsf{mul}}_{\mathsf{kv}'} T$ then case (3a) applies for $s >_{\mathsf{KV}'} t$. Otherwise, $S \geq^{\mathsf{mul}}_{\mathsf{kv}'} T$ implies $n = m$ and thus $|\nabla_\circ(s)| < |\nabla_\circ(t)|$. Hence neither case (3b) nor (3c) applies.

[6]This property is well-known for standard multiset extensions (involving a single strict order). It is also not difficult to prove for the multiset extension defined in Definition 2.6.

We reduce a non-empty CNF SAT problem $\phi = \{C_1, \ldots, C_m\}$ over propositional variables $x_1, \ldots, x_n$ to the decision problem $S_\phi >^{\mathsf{mul}}_{\mathsf{kv}'} T_\phi$. The multisets S_ϕ and T_ϕ will consist of terms in $\mathcal{T}(\{\mathsf{a}, \mathsf{f}\}, \{x_1, \ldots, x_n, y_1, \ldots, y_m\})$, where a is a constant with $w(\mathsf{a}) = w_0$ and f has arity $m + 1$. For each $j \in \{1, \ldots, m\}$ and literal l, we define

$$s_j(l) = \begin{cases} y_j & \text{if } l \in C_j \\ \mathsf{a} & \text{otherwise} \end{cases}$$

Moreover, for each $i \in \{1, \ldots, n\}$ we define

$$\begin{aligned} t_i^+ &= \mathsf{f}(x_i, s_1(x_i), \ldots, s_m(x_i)) \\ t_i^- &= \mathsf{f}(x_i, s_1(\neg x_i), \ldots, s_m(\neg x_i)) \\ t_i &= \mathsf{f}(x_i, \mathsf{a}, \ldots, \mathsf{a}) \end{aligned}$$

Note that $w(t_i^+) = w(t_i^-) = w(t_i) > w(y_j)$ for all $i \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, m\}$. Finally, we define

$$S_\phi = \{t_1^+, t_1^-, \ldots, t_n^+, t_n^-\} \qquad\qquad T_\phi = \{t_1, \ldots, t_n, y_1, \ldots, y_m\}$$

Note that for every $i \in \{1, \ldots, n\}$ there is no $s \in S_\phi$ such that $s >_{\mathsf{kv}} t_i$. Hence $S_\phi >^{\mathsf{mul}}_{\mathsf{kv}'} T_\phi$ iff S_ϕ can be written as $\{s_1, \ldots, s_n, s'_1, \ldots, s'_n\}$ such that $s_i \geq_{\mathsf{kv}'} t_i$ for all $i \in \{1, \ldots, n\}$, and for all $j \in \{1, \ldots, m\}$ there exists an $1 \leq i \leq n$ such that $s'_i >_{\mathsf{kv}} y_j$. It is easy to see that the only candidates for s_i are t_i^+ and t_i^-.

Now suppose $S_\phi >^{\mathsf{mul}}_{\mathsf{kv}'} T_\phi$ with S_ϕ written as above. Consider the assignment α defined as follows: $\alpha(x_i)$ is true iff $s_i = t_i^-$. We claim that α satisfies every $C_j \in \phi$. We know that there exists $i \in \{1, \ldots, n\}$ such that $s'_i >_{\mathsf{kv}} y_j$ and thus also $y_j \in \mathsf{Var}(s'_i)$. This is only possible if $x_i \in C_j$ (when $s'_i = t_i^+$) or $\neg x_i \in C_j$ (when $s'_i = t_i^-$). Hence, by construction of α, α satisfies C_j.

Conversely, suppose α satisfies ϕ. Let $s'_i = t_i^+$ and $s_i = t_i^-$ if $\alpha(x_i)$ is true and $s'_i = t_i^-$ and $s_i = t_i^+$ if $\alpha(x_i)$ is false. We trivially have $s_i \geq_{\mathsf{kv}'} t_i$ for all $i \in \{1, \ldots, n\}$. Moreover, for each $j \in \{1, \ldots, m\}$, C_j contains a literal $l = (\neg)x_i$ such that $\alpha(l)$ is true. By construction, $y_j \in \mathsf{Var}(s'_i)$ and thus $s'_i >_{\mathsf{kv}} y_j$. Since ϕ is non-empty, $m > 0$ and hence $S_\phi >^{\mathsf{mul}}_{\mathsf{kv}'} T_\phi$ as desired. □

8.6.2 Membership Problems for $>_{\mathsf{ACKBO}}$ and $>_{\mathsf{KV}}$

In this section, we investigate complexity of membership problems for $>_{\mathsf{KV}}$ and $>_{\mathsf{ACKBO}}$. While the corresponding problem for $>_{\mathsf{KV}'}$ is NP-hard, it turns out that these problems are polynomially decidable.

Let S be a sequence of terms $s_1, \ldots, s_n$. We denote by $S[t]_i$ the sequence obtained by replacing s_i with t in S, and by $S[\,]_i$ the sequence obtained by removing s_i from S. Moreover, we write $\{S\}$ as a shorthand for the multiset $\{s_1, \ldots, s_n\}$.

Lemma 8.40 *Let $\langle \succsim, \succ \rangle$ be an order pair on terms such that $\sim := \succsim \setminus \succ$ is symmetric, and let S and T denote sequences of terms. If $s \sim t$, then $\{S\} \uplus \{s\} \succ^{\mathsf{mul}} \{T\} \uplus \{t\}$ iff $\{S\} \succ^{\mathsf{mul}} \{T\}$.*

Proof. The "if" direction is trivial. For the "only if" direction, we assume $S[s]_i \sqsupset_k^{\mathsf{mul}} T[t]_j$ and show $\{S[\,]_i\} \succ^{\mathsf{mul}} \{T[\,]_j\}$. We distinguish four cases.

- If $i, j \leq k$ then $s_j \succsim t_j = t \sim s = s_i \succsim t_i$ and hence $S[s_j]_i[\,]_j \sqsupset_{k-1}^{\mathsf{mul}} T[t_i]_j[\,]_i$. Since $\{S[s_j]_i[\,]_j\} = \{S[\,]_i\}$ and $\{T[t_i]_j[\,]_i\} = \{T[\,]_j\}$, we are done.
- If $i \leq k < j$ then $s_l \succ t_j = t \sim s = s_i \succsim t_i$ for some $l > k$. Hence $S[\,]_i \sqsupset_{k-1}^{\mathsf{mul}} T[t_i]_j[\,]_i$.
- If $j \leq k < i$ then $s_j \succsim t_j = t \sim s = s_i$ and thus $s_j \succ t_l$ for every $l > k$ such that $s_i \succ t_l$. Hence $S[s_j]_i[\,]_j \sqsupset_{k-1}^{\mathsf{mul}} T[\,]_j$.
- The remaining case $k < i, j$ is analogous to the previous case, and we obtain $S[\,]_i \sqsupset_k^{\mathsf{mul}} T[\,]_j$. □

Lemma 8.41 *Let $\langle \succsim, \succ \rangle$ be an order pair such that $\sim := \succsim \setminus \succ$ is symmetric and the decision problems for $\succsim$ and $\succ$ are in P. Then, the decision problem for $\succ^{\mathsf{mul}}$ is in P.*

Proof. Suppose we want to decide whether two multisets S and T satisfy $S \succ^{\mathsf{mul}} T$. We first check if there exists a pair $(s, t) \in S \times T$ such that $s \sim t$, which can be done by testing $s \succsim t$ and $s \not\succ t$ at most $|S| \times |T|$ times. If such a pair is found then according to Lemma 8.40, the problem is reduced to $S - \{s\} \succ^{\mathsf{mul}} T - \{t\}$. Otherwise, we check for each $t \in T$ whether there exists $s \in S$ such that $s \succ t$, which can be done by testing $s \succ t$ at most $|S| \times |T|$ times. □

Using the above lemma, we obtain the following result by a straightforward induction argument.

Corollary 8.42 *The decision problems for $>_{\mathsf{ACKBO}}$ and $>_{\mathsf{KV}}$ are in P.* □

8.6.3 Orientability Problem for $>_{\mathsf{KV}}$

It is well-known [45] that KBO orientability is decidable in polynomial time. In this section, we show that the corresponding problem for $>_{\mathsf{KV}}$ is NP-hard even for ground TRSs. The same result will be obtained for $>_{\mathsf{ACKBO}}$ in Section 8.6.4.

To this end, we reduce a SAT instance to an orientability problem; let $\phi = \{C_1, \ldots, C_n\}$ be a CNF SAT problem over propositional variables $p_1, \ldots, p_m$. We consider the signature $\mathcal{F}_\phi$ consisting of an AC symbol $+$, constants c and $\mathsf{d}_1, \ldots, \mathsf{d}_n$, and unary function symbols $p_1, \ldots, p_m$, a, b, and e_i^j for all $i \in \{1, \ldots, n\}$ and $j \in \{0, \ldots, m\}$. We define a ground TRS $\mathcal{R}_\phi$ on $\mathcal{T}(\mathcal{F}_\phi)$ such that $>_{\mathsf{KV}}$ orients $\mathcal{R}_\phi$ iff ϕ is satisfiable. The TRS $\mathcal{R}_\phi$ will contain the following base system $\mathcal{R}_0$ that enforces certain constraints on the precedence and the weight function:

$$\begin{array}{cccc} \mathsf{a}(\mathsf{c}+\mathsf{c}) \to \mathsf{a}(\mathsf{c})+\mathsf{c} & \mathsf{b}(\mathsf{c})+\mathsf{c} \to \mathsf{b}(\mathsf{c}+\mathsf{c}) & & \mathsf{a}(\mathsf{b}(\mathsf{b}(\mathsf{c}))) \to \mathsf{b}(\mathsf{a}(\mathsf{a}(\mathsf{c}))) \\ \mathsf{a}(\mathsf{a}(\mathsf{c})) \to \mathsf{b}(p_1(\mathsf{c})) & \mathsf{a}(p_1(\mathsf{c})) \to \mathsf{b}(p_2(\mathsf{c})) & \cdots & \mathsf{a}(p_m(\mathsf{c})) \to \mathsf{b}(\mathsf{a}(\mathsf{c})) \end{array}$$

Lemma 8.43 *The order $>_{\mathsf{KV}}$ is compatible with $\mathcal{R}_0$ iff $\mathsf{a} > + > \mathsf{b}$ and $w(\mathsf{a}) = w(\mathsf{b}) = w(p_j)$ for all $j \in \{1, \ldots, m\}$.* $\square$

Definition 8.44 *Consider the clause C_i of the form $\{p'_1, \ldots, p'_k, \neg p''_1, \ldots, \neg p''_l\}$. Let U, U', V, and W denote the followings multisets:*

$$\begin{array}{ll} U = \{p'_1(\mathsf{b}(\mathsf{d}_i)), \ldots, p'_k(\mathsf{b}(\mathsf{d}_i))\} & V = \{p''_0(\mathsf{e}_i^{0,1}), \ldots, p''_{l-1}(\mathsf{e}_i^{l-1,l}), p''_l(\mathsf{e}_i^{l,0})\} \\ U' = \{\mathsf{b}(p'_1(\mathsf{d}_i)), \ldots, \mathsf{b}(p'_k(\mathsf{d}_i))\} & W = \{p''_0(\mathsf{e}_i^{0,0}), \ldots, p''_l(\mathsf{e}_i^{l,l})\} \end{array}$$

where we write p''_0 for a and $\mathsf{e}_i^{j,k}$ for $\mathsf{e}_i^j(\mathsf{e}_i^k(\mathsf{c}))$. The TRS $\mathcal{R}_\phi$ is defined as the union of $\mathcal{R}_0$ and $\{\ell_i \to r_i \mid 1 \le i \le n\}$ with

$$\ell_i = \mathsf{b}(\mathsf{b}(\mathsf{c}+\mathsf{c})) + \textstyle\sum U + \sum V \qquad r_i = \mathsf{b}(\mathsf{c}) + \mathsf{b}(\mathsf{c}) + \sum U' + \sum W$$

Note that the symbols d_i and $\mathsf{e}_i^0, \ldots, \mathsf{e}_i^l$ are specific to the rule $\ell_i \to r_i$.

Lemma 8.45 *Let $\mathsf{a} > + > \mathsf{b}$. Then, $\mathcal{R}_\phi \subseteq >_{\mathsf{KV}}$ for some $\langle w, w_0 \rangle$ iff for every i there is some p such that $p \in C_i$ with $p \not< +$ or $\neg p \in C_i$ with $+ > p$.*

Proof. For the "if" direction we reason as follows. Consider a (partial) weight function w such that $w(\mathsf{a}) = w(\mathsf{b}) = w(p_j)$ for every $j \in \{1, \ldots m\}$. We obtain $\mathcal{R}_0 \subseteq >_{\mathsf{KV}}$ from Lemma 8.43. Further, consider $C_i = \{p'_1, \ldots, p'_k, \neg p''_1, \ldots, \neg p''_l\}$ and ℓ_i, r_i, U, V and W defined above. Let $L = \nabla_+(\ell_i)$ and $R = \nabla_+(r_i)$. We clearly have $L{\restriction}_+^{\not<} = U{\restriction}_+^{\not<} \cup V{\restriction}_+^{\not<}$ and $R{\restriction}_+^{\not<} = W{\restriction}_+^{\not<}$. It is easy to show that $w(\ell_i) = w(r_i)$. We show $\ell_i >_{\mathsf{KV}} r_i$ by distinguishing two cases.

1. First suppose that $p'_j \not< +$ for some $j \in \{1,\dots,k\}$. We have $p'_j(\mathsf{b}(\mathsf{d}_i)) \in U\restriction_{+}^{\not<}$. Extend the weight function w such that

$$w(\mathsf{d}_i) = 1 + 2 \cdot \max\{w(\mathsf{e}_i^0),\dots,w(\mathsf{e}_i^l)\}$$

Then $p'_j(\mathsf{b}(\mathsf{d}_i)) >_{\mathsf{kv}} t$ for all terms $t \in W$ and hence $L\restriction_{+}^{\not<} >_{\mathsf{kv}}^{\mathsf{mul}} R\restriction_{+}^{\not<}$. Therefore $\ell_i >_{\mathsf{KV}} r_i$ by case (3a).

2. Otherwise, $U\restriction_{+}^{\not<} = \varnothing$ holds. By assumption $+ > p''_j$ for some $j \in \{1,\dots,l\}$. Consider the smallest m such that $+ > p''_m$. Extend the weight function w such that

$$w(\mathsf{e}_i^m) = 1 + 2 \cdot \max\{w(\mathsf{e}_i^j) \mid j \neq m\}$$

Then $w(p''_{m-1}(\mathsf{e}_i^{m-1,m})) > w(p''_j(\mathsf{e}_i^{j,j}))$ for all $j \neq m$. From $p''_{m-1} > +$ we infer $p''_{m-1}(\mathsf{e}_i^{m-1,m}) \in V\restriction_{+}^{\not<}$. (Note that $p''_{m-1} = \mathsf{a} > +$ if $m = 1$.) By definition of m, $p''_m(\mathsf{e}_i^{m,m}) \notin W\restriction_{+}^{\not<}$. It follows that $L\restriction_{+}^{\not<} >_{\mathsf{kv}}^{\mathsf{mul}} R\restriction_{+}^{\not<}$ and thus $\ell_i >_{\mathsf{KV}} r_i$ by case (3a).

Next we prove the "only if" direction. So suppose there exists a weight function w such that $\mathcal{R}_\phi \subseteq >_{\mathsf{KV}}$. We obtain $w(\mathsf{a}) = w(\mathsf{b}) = w(p_j)$ for all $j \in \{1,\dots,m\}$ from Lemma 8.43. It follows that $w(\ell_i) = w(r_i)$ for every $C_i \in \phi$. Suppose for a proof by contradiction that there exists $C_i \in \phi$ such that $+ > p$ for all $p \in C_i$ and $p \not< +$ whenever $\neg p \in C_i$. So $L\restriction_{+}^{\not<} = V$ and $R\restriction_{+}^{\not<} = W$. Since $|R| = |L| + 1$, we must have $\ell_i >_{\mathsf{KV}} r_i$ by case (3a) and thus $V >_{\mathsf{kv}} W$. Let s be a term in V of maximal weight. We must have $w(s) \geq w(t)$ for all terms $t \in W$. By construction of the terms in V and W, this is only possible if all symbols e_i^j have the same weight. It follows that all terms in V and W have the same weight. Since $|V| = |W|$ and for every term $s' \in V$ there exists a unique term $t' \in W$ with $\mathsf{root}(s') = \mathsf{root}(t')$, we conclude $V =_{\mathsf{kv}} W$, which provides the desired contradiction. □

After these preliminaries we are ready to prove NP-hardness.

Theorem 8.46 *The (ground) orientability problem for $>_{\mathsf{KV}}$ is NP-hard.*

Proof. It is sufficient to prove that a CNF formula $\phi = \{C_1,\dots,C_n\}$ is satisfiable iff the corresponding $\mathcal{R}_\phi$ is orientable by $>_{\mathsf{KV}}$. Note that the size of $\mathcal{R}_\phi$ is linear in the size of ϕ. First suppose that ϕ is satisfiable. Let α be a satisfying assignment for the atoms $p_1,\dots,p_m$. Define the precedence $>$ as follows: $\mathsf{a} > + > \mathsf{b}$ and $p_j > +$ if $\alpha(p_j)$ is true and $+ > p_j$ if $\alpha(p_j)$ is false. Then $\mathcal{R}_\phi \subseteq >_{\mathsf{KV}}$ follows from Lemma 8.45. Conversely, if $\mathcal{R}_\phi$ is compatible with $>_{\mathsf{KV}}$ then we define an assignment α for the atoms in ϕ as follows: $\alpha(p)$ is true if $p \not< +$ and $\alpha(p)$ is false if $+ > p$. We claim that α satisfies ϕ. Let C_i be a clause in ϕ. According to Lemma 8.45, $p \not< +$ for one of the atoms p in C_i or $+ > p$ for one of the negative literals $\neg p$ in C_i. Hence α satisfies C_i by definition. □

8.6.4 Orientability Problem for $>_{\mathsf{ACKBO}}$

Now we adapt the reasoning in the previous section to prove NP-hardness of the orientability problem for $>_{\mathsf{ACKBO}}$. To this end, we introduce the TRS $\mathcal{R}_0'$ consisting of the rules

$$\mathsf{a}(p_1(\mathsf{c})) \to p_1(\mathsf{a}(\mathsf{c})) \qquad \cdots \qquad \mathsf{a}(p_m(\mathsf{c})) \to p_m(\mathsf{a}(\mathsf{c}))$$

together with a rule $\mathsf{e}_i^0(\mathsf{e}_i^1(\mathsf{c})) \to \mathsf{e}_i^1(\mathsf{e}_i^0(\mathsf{c}))$ for each clause C_i that contains a negative literal. The next property is immediate.

Lemma 8.47 *If $\mathcal{R}_0' \subseteq >_{\mathsf{ACKBO}}$ then $\mathsf{e}_i^0 > \mathsf{e}_i^1$ for all $i \in \{1,\ldots,n\}$ and $\mathsf{a} > p_j$ for all $j \in \{1,\ldots,m\}$.* □

We denote the TRS $\mathcal{R}_0 \cup \mathcal{R}_0' \cup \{\ell_i \to r_i \mid 1 \le i \le n\}$ by $\mathcal{R}_\phi'$.

Lemma 8.48 *Suppose $\mathsf{a} > + > \mathsf{b}$ and the consequence of Lemma 8.47 holds. Then $\mathcal{R}_\phi' \subseteq >_{\mathsf{ACKBO}}$ for some $\langle w, w_0\rangle$ iff for every i there is some p such that $p \in C_i$ with $p \not< +$ or $\neg p \in C_i$ with $+ > p$.*

Proof. The "if" direction is analogous to Lemma 8.45. Let us prove the "only if" direction by contradiction. Suppose that $+ > p_j'$ for every $j \in \{1,\ldots,k\}$, $p_j'' \not< +$ for every $j \in \{1,\ldots,l\}$, and $\mathcal{R}_\phi' \subseteq >_{\mathsf{ACKBO}}$. As discussed in the proof of Lemma 8.45, for the multisets V and W of Definition 8.44 we obtain $V >_{\mathsf{ACKBO}}^{\mathsf{mul}} W$ and all terms in V and W have the same weight. With help of Lemma 8.47 we infer that $\mathsf{a}(\mathsf{e}_i^0(\mathsf{e}_i^0(\mathsf{c}))) \in W$ is greater than every other term in V and W. This contradicts $V >_{\mathsf{ACKBO}}^{\mathsf{mul}} W$. □

Now the desired result follows immediately.

Corollary 8.49 *The ground orientability problem for $>_{\mathsf{ACKBO}}$ is NP-hard.* □

The overview of complexity results are presented in Table 8.1.

Table 8.1: Complexity results (KV is the ground version of $>_{\mathsf{KV}}$).

problem	KBO	AC-KBO	KV	KV′
membership	P	P	P	NP-hard
orientability	P	NP-hard	NP-hard	NP-hard

8.7 Subterm Coefficients

Subterm coefficients were introduced by Ludwig and Waldmann [58] in order to cope with rewrite rules like $\mathsf{f}(x) \to \mathsf{g}(x, x)$ which violate the variable condition. A *subterm coefficient function* is a partial mapping $sc\colon \mathcal{F} \times \mathbb{N} \to \mathbb{N}$ such that for a function symbol f of arity n we have $sc(f, i) > 0$ for all $i \in \{1, \ldots, n\}$. Given a weight function $\langle w, w_0\rangle$ and a subterm coefficient function sc, the weight of a term is inductively defined as follows:

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

The *variable coefficient* $\mathsf{vc}(x, t)$ of a variable x in a term t is inductively defined as follows: $\mathsf{vc}(x, t) = 1$ if $t = x$, $\mathsf{vc}(x, t) = 0$ if $t \in \mathcal{V} \setminus \{x\}$, and $\mathsf{vc}(x, f(t_1, \ldots, t_n)) = sc(f, 1) \cdot \mathsf{vc}(x, t_1) + \cdots + sc(f, n) \cdot \mathsf{vc}(x, t_n)$.

Definition 8.50 *The order $>^{sc}_{\mathsf{ACKBO}}$ is obtained from Definition 8.19 by replacing the condition "$|s|_x \geq |t|_x$ for all $x \in \mathcal{V}$" with "$\mathsf{vc}(x, s) \geq \mathsf{vc}(x, t)$ for all $x \in \mathcal{V}$" and using the modified weight function introduced above.*

In order to guarantee AC compatibility of $>^{sc}_{\mathsf{ACKBO}}$, the subterm coefficient function sc has to assign the value 1 to arguments of AC symbols. This follows by considering the terms $t \circ (u \circ v)$ and $(t \circ u) \circ v$ for an AC symbol $\circ$ with $sc(\circ, 1) = m$ and $sc(\circ, 2) = n$. We have

$$\begin{aligned} w(t \circ (u \circ v)) &= 2 \cdot w(\circ) + m \cdot w(t) + mn \cdot w(u) + n^2 \cdot w(v) \\ w((t \circ u) \circ v) &= 2 \cdot w(\circ) + m^2 \cdot w(t) + mn \cdot w(u) + n \cdot w(v) \end{aligned}$$

Since $w(t \circ (u \circ v)) = w((t \circ u) \circ v)$ must hold for all possible terms t, u, and v, it follows that $m = m^2$ and $n^2 = n$, implying $m = n = 1$.[7] The proof of the following theorem is very similar to the one of Theorem 8.20 and hence omitted.

[7]This condition is also obtained by restricting [7, Proposition 4] to linear polynomials.

Theorem 8.51 *If $sc(f,1) = sc(f,2) = 1$ for every function symbol $f \in \mathcal{F}_{\mathsf{AC}}$ then $>^{sc}_{\mathsf{ACKBO}}$ is an AC-compatible simplification order.* □

Example 8.52 *Consider the following TRS $\mathcal{R}$ with $\mathsf{f} \in \mathcal{F}_{\mathsf{AC}}$:*

$$\mathsf{g}(\mathsf{0}, \mathsf{f}(x, x)) \to x \tag{8.3}$$

$$\mathsf{g}(x, \mathsf{s}(y)) \to \mathsf{g}(\mathsf{f}(x, y), \mathsf{0}) \tag{8.4}$$

$$\mathsf{g}(\mathsf{s}(x), y) \to \mathsf{g}(\mathsf{f}(x, y), \mathsf{0}) \tag{8.5}$$

$$\mathsf{g}(\mathsf{f}(x, y), \mathsf{0}) \to \mathsf{f}(\mathsf{g}(x, \mathsf{0}), \mathsf{g}(y, \mathsf{0})) \tag{8.6}$$

Termination of $\mathcal{R}$ was shown using AC dependency pairs in [50, Example 4.2.30]. Consider a precedence $\mathsf{g} > \mathsf{f} > \mathsf{s} > \mathsf{0}$, and weights and subterm coefficients given by $w_0 = 1$ and the following interpretation $\mathcal{A}$, mapping function symbols in $\mathcal{F}$ to linear polynomials over $\mathbb{N}$:

$$\mathsf{s}_{\mathcal{A}}(x) = x + 6 \qquad \mathsf{g}_{\mathcal{A}}(x, y) = 4x + 4y + 5 \qquad \mathsf{f}_{\mathcal{A}}(x, y) = x + y + 3 \qquad \mathsf{0}_{\mathcal{A}} = 1$$

It is easy to check that the first three rules result in a weight decrease. The left- and right-hand side of rule (8.6) are both interpreted as $4x + 4y + 21$, so both terms have weight 29, but since $\mathsf{g} > \mathsf{f}$ we conclude termination of $\mathcal{R}$ from case 1 in Definition 8.19 (8.50). Note that termination of $\mathcal{R}$ cannot be shown by AC-RPO or any of the previously considered versions of AC-KBO.

8.8 Experiments

We ran experiments on a server equipped with eight dual-core AMD Opteron® processors 885 running at a clock rate of 2.6GHz with 64GB of main memory. The different versions of AC-KBO considered in this chapter as well as AC-RPO [65] were implemented on top of $\mathsf{T_TT_2}$ using encodings in SAT/SMT. These encodings resemble those for standard KBO [91] and transfinite KBO [81]. The encoding of multiset extensions of order pairs are based on [12], but careful modifications were required to deal with submultisets induced by the precedence.

For termination experiments, our test set comprises all AC problems in the *Termination Problem Data Base*,[8] all examples in this chapter, some further problems harvested from the literature, and constraint systems produced by the completion tool mkbtt [79] (145 TRSs in total). The timeout was set to 60 seconds. The results are summarized in Table 8.2, where we list for each order

[8]`http://termination-portal.org/wiki/TPDB`

Table 8.2: Experiments on 145 termination problems.

	orientability			AC-DP		
method	yes	T.O.	time	yes	T.O.	time
AC-KBO	32	0	1.7	66	3	463.1
Steinbach	23	0	1.6	50	2	463.2
Korovin & Voronkov	30	0	2.0	66	4	474.3
KV′	30	0	2.1	66	3	472.4
subterm coefficients	37	0	47.1	68	2	464.7
AC-RPO	63	0	2.8	79	4	501.5
total	72			94		

Table 8.3: Experiments on 67 completion problems.

method	success	T.O.	time
AC-KBO	25	37	2278.6
Steinbach	24	36	2235.4
Korovin & Voronkov	25	37	2279.4
KV′	25	37	2279.6
subterm coefficients	28	26	1724.7
AC-RPO	28	26	1701.6
total	31		

the number of successful termination proofs, the total time, and the number of timeouts (T.O.). The 'orientability' column directly applies the order to orient all the rules. Although AC-RPO succeeds on more input problems, termination of 9 TRSs could only be established by (variants of) AC-KBO. We found that our definition of AC-KBO is about equally powerful as Korovin and Voronkov's order, but both are considerably more useful than Steinbach's version. When it comes to proving termination, we did not observe a difference between Definitions 8.13 and 8.15. Subterm coefficients clearly increase the success rate, although efficiency is affected. In all settings partial precedences were allowed.

The 'AC-DP' column applies the order in the AC-dependency pair framework of [1], in combination with *argument filterings* and *usable rules*. Here AC symbols in dependency pairs are *unmarked*, as proposed by Marché and Urbain [60] in [60]. In this setting the variants of AC-KBO become considerably more powerful and competitive to AC-RPO, since argument filterings relax the variable condition, as pointed out Zankl et al. [91].

For completion experiments, we ran the normalized completion tool mkbtt with AC-RPO and the variants of AC-KBO for termination checks on 67 equational systems collected from the literature. The overall timeout was set to 60 seconds, the timeout for each termination check to 1.5 seconds. Table 8.3 summarizes our results, listing for each order the number of successful completions, the total time, and the number of timeouts. All experimental details, source code, and $\mathsf{T_TT_2}$ binaries are available online.[9]

We close this chapter with an example that can be completed using AC-KBO, whereas AC-RPO does not succeed.

Example 8.53 *Consider the following TRS $\mathcal{R}$ [60] for addition of binary numbers:*

$$
\begin{array}{lll}
\# + \mathsf{0} \to \# & x\mathsf{0} + y\mathsf{0} \to (x+y)\mathsf{0} & x\mathsf{1} + y\mathsf{1} \to (x+y+\#\mathsf{1})\mathsf{0} \\
x + \# \to x & x\mathsf{0} + y\mathsf{1} \to (x+y)\mathsf{1} &
\end{array}
$$

Here $+ \in \mathcal{F}_{\mathsf{AC}}$, $\mathsf{0}$ and $\mathsf{1}$ are unary operators in postfix notation, and $\#$ denotes the empty bit sequence. For example, $\#\mathsf{100}$ represents the number 4. This TRS is not compatible with AC-RPO but AC termination can easily be shown by AC-KBO, for instance with the weight function $\langle w, w_0\rangle$ with $w(\mathsf{+}) = 0$, $w_0 = w(\mathsf{0}) = w(\#) = 1$, and $w(\mathsf{1}) = 3$. The system can be completed into an AC convergent TRS using AC-KBO but not with AC-RPO.

[9]`http://cl-informatik.uibk.ac.at/software/ackbo`

Chapter 9

Conclusion

In this thesis, we have considered unifying existing techniques for proving termination of term rewrite systems. In Chapter 3, we summarized the variants of the order of Knuth and Bendix [43]. There we presented a modification to the generalized KBO of Middeldorp and Zantema [62] in order to properly generalize the original KBO.

In Chapter 4, we introduced WPO that generalize not only KBO but also most of the well-known reduction orders. As instances of WPO, we presented the following orders: WPO($\mathcal{S}um$) that subsumes KBO, WPO($\mathcal{P}ol$) that subsumes POLO and TKBO, WPO($\mathcal{M}ax$) that subsumes LPO, WPO($\mathcal{MS}um$) that unifies KBO and LPO, and WPO($\mathcal{MP}ol$) that unifies all of them. In Chapter 5, we introduced a partial status for WPO to obtain a reduction pair, and presented further refinements. We have shown that as a reduction pair, WPO subsumes KBO, LPO and TKBO with argument filters, POLO, and matrix interpretations. In Chapter 6, we also presented SMT encodings for these techniques. The orientability problems of WPO($\mathcal{S}um$), WPO($\mathcal{M}ax$) and WPO($\mathcal{MS}um$) are decidable, since they are reduced to satisfiability problems of linear integer arithmetic which is known to be decidable.

In Chapter 7, we described the implementation and techniques of the termination tool NaTT, the first tool that implements WPO. The implementation of WPO is described in detail, and some techniques for cooperating SMT solvers are presented. Together with these efforts, NaTT is one of the most efficient and strongest tools for proving termination of TRSs. Through experiments using NaTT, we verified the significance of the proposed techniques of this thesis.

In Chapter 8, we revisited the two variants of AC-compatible extensions of KBO. We extended the first version $>_{\mathsf{S}}$ introduced by Steinbach [70] to a new version $>_{\mathsf{ACKBO}}$, and presented a rigorous correctness proof. By this we conclude correctness of $>_{\mathsf{S}}$, which had been put in doubt in [46]. We also modified the order $>_{\mathsf{KV}}$ by Korovin and Voronkov [46] to a new version $>_{\mathsf{KV}'}$ which is monotone on non-ground terms, in contrast to $>_{\mathsf{KV}}$. We also presented several complexity results regarding these variants. While a polynomial time algorithm is known for the orientability problem of standard KBO [45], the problem becomes NP-hard even for the ground version of $>_{\mathsf{KV}}$, as well as for our $>_{\mathsf{ACKBO}}$. Somewhat unexpectedly, even deciding $>_{\mathsf{KV}'}$ is NP-hard while it is linear for standard KBO [57]. In contrast, the corresponding problem is polynomial-time for our $>_{\mathsf{ACKBO}}$. Finally, we implemented these variants of AC-compatible KBO as well as the AC-dependency pair framework of Alarcón et al. [1]. We presented full experimental results both for termination proving and normalized completion.

Future Works

When combining $\mathsf{WPO}(\mathcal{P}ol)$ and $\mathsf{WPO}(\mathcal{M}ax)$, we only considered a straight-forward method using 'weight statuses', and moreover heuristically fixed the weight status. We leave it for future work to search for other possible weight statuses, or to find more sophisticated combinations of max-polynomials such as $f_{\mathcal{A}}(x, y, z) = x + \max(y, z)$, or even trying other algebras including *ordinal interpretations* [81, 83]. For efficiency, real arithmetic is also attractive to consider, since SMT for real arithmetic is often more efficient than for integer arithmetic. To this end, we will have to reconstruct the proof of well-foundedness of WPO, since our current proof relies on well-foundedness of the underlying order, which does not hold anymore for real numbers.

Another obvious future work is to combine results of Chapter 8 with WPO to introduce an AC-compatible variant of WPO. Moreover, it might be also interesting to extend WPO for higher-order case. Since RPOLO has strength in its higher-order version [8], we expect their technique can be extended for WPO.

Because of its efficiency, NaTT is especially strong on larger systems. In general, a larger input TRS requires a larger proof script to be produced, which is quite difficult to be checked by hand. Thus we leave it for future work to produce proofs in the *certifiable proof format*.[1]

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

Bibliography

[1] B. Alarcón, S. Lucas, and J. Meseguer. A dependency pair framework for A$\vee$C-termination. In *Proceedings of 8th International Workshop on Rewriting Logic and its Applications (WRLA '10)*, Volume 6381 of Lecture Notes in Computer Science, pages 36–52, 2010.

[2] T. Arts and J. Giesl. Termination of term rewriting using dependency pairs. *Theoretical Computer Science*, 236(1-2):133–178, 2000.

[3] F. Baader and T. Nipkow. *Term Rewriting and All That.* Cambridge University Press, 1998.

[4] L. Bachmair. Assosiative-commutative reduction orderings. *Information Processing Letters*, 43:21–27, 1992.

[5] L. Bachmair and D. A. Plaisted. Termination orderings for associative-commutative rewriting systems. *Journal of Symbolic Computation*, 1:329–349, 1985.

[6] A. M. Ben-Amram and M. Codish. A SAT-based approach to size change termination with global ranking functions. In *Proceedings of 14th Tools and Algorithms for the Construction and Analysis of Systems (TACAS '08)*, Volume 4963 of Lecture Notes in Computer Science, pages 218–232, 2008.

[7] A. Ben Cherifa and P. Lescanne. Termination of rewriting systems by polynomial interpretations and its implementation. *Science of Computer Programming*, 9(2):137–159, 1987.

[8] M. Bofill, C. Borralleras, E. Rodríguez-Carbonell, and A. Rubio. The recursive path and polynomial ordering for first-order and higher-order terms. *Journal of Logic and Computation*, 23(1):263–305, 2013.

[9] C. Borralleras, M. Ferreira, and A. Rubio. Complete monotonic semantic path orderings. In *Proceedings of 17th International Conference on Automated Deduction (CADE '00)*, Volume 1831 of Lecture Notes in Computer Science, pages 346–364, 2000.

[10] C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio. Solving non-linear polynomial arithmetic via SAT modulo linear arithmetic. In *Proceedings of 22nd International Conference on Automated Deduction (CADE '09)*, Volume 5663 of Lecture Notes in Computer Science, pages 294–305, 2009.

[11] C. Borralleras, S. Lucas, R. Navarro-Marset, E. Rodríguez-Carbonell, and A. Rubio. SAT modulo linear arithmetic for solving polynomial constraints. *Journal of Automated Reasoning*, 48(1):107–131, 2012.

[12] M. Codish, J. Giesl, P. Schneider-Kamp, and R. Thiemann. SAT solving for termination proofs with recursive path orders and dependency pairs. *Journal of Automated Reasoning*, 49(1):53–93, 2012.

[13] E. Contejean, C. Marché, B. Monate, and X. Urbain. Proving termination of rewriting with CiME. In *6th International Workshop on Termination (WST '03)*, pages 71–73, 2003.

[14] N. Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982.

[15] N. Dershowitz. Termination of rewriting. *Journal of Symbolic Computation*, 3(1–2):69–115, 1987.

[16] N. Dershowitz. Termination by abstraction. In *Proceedings of 20th International Conference on Logic Programming (ICLP '04)*, Volume 3132 of Lecture Notes in Computer Science, pages 1–18, 2004.

[17] N. Dershowitz and C. Hoot. Natural termination. *Theoretical Computer Science*, 142(2):179–207, 1995.

[18] N. Dershowitz, J. Hsiang, A. Josephson, and D. Plaisted. Associative-commutative rewriting. In *Proceedings of 8th International Joint Conference on Artificial Intelligence (IJCAI '83)*, pages 940–944, 1983.

[19] J. Dick, J. Kalmus, and U. Martin. Automating the Knuth Bendix ordering. *Acta Infomatica*, 28:95–119, 1990.

[20] J. Endrullis, J. Waldmann, and H. Zantema. Matrix interpretations for proving termination of term rewriting. *Journal of Automated Reasoning*, 40 (2-3):195–220, 2008.

[21] S. Falke and D. Kapur. A term rewriting approach to the automated termination analysis of imperative programs. In *Proceedings of 22nd International Conference on Automated Deduction (CADE '09)*, Volume 5663 of Lecture Notes in Artificial Intelligence, pages 277–293, 2009.

[22] S. Falke, D. Kapur, and C. Sinz. Termination analysis of C programs using compiler intermediate languages. In *Proceedings of 22nd International Conference on Rewriting Techniques and Applications (RTA '11)*, Volume 10 of Leibniz International Proceedings in Informatics, pages 41–50, 2011.

[23] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. SAT solving for termination analysis with polynomial interpretations. In *Proceedings of 10th Theory and Applications of Satisfiability Testing (SAT '07)*, Volume 4501 of Lecture Notes in Computer Science, pages 340–354, 2007.

[24] C. Fuhs, J. Giesl, A. Middeldorp, P. Schneider-Kamp, R. Thiemann, and H. Zankl. Maximal termination. In *Proceedings of 19th International Conference on Rewriting Techniques and Applications (RTA '08)*, Volume 5117 of Lecture Notes in Computer Science, pages 110–125, 2008.

[25] A. Geser. An improved general path order. *Applicable Algebra in Engineering, Communication and Computing*, 7(6):469–511, 1996.

[26] J. Giesl and D. Kapur. Dependency pairs for equational rewriting. In *Proceedings of 12th International Conference on Rewriting Techniques and Applications (RTA '01)*, Volume 2051 of Lecture Notes in Computer Science, pages 93–107, 2001.

[27] J. Giesl, R. Thiemann, and P. Schneider-Kamp. The dependency pair framework: Combining techniques for automated termination proofs. In *Proceedings of 11th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '04)*, Volume 3452 of Lecture Notes in Artificial Intelligence, pages 75–90, 2004.

[28] J. Giesl, R. Thiemann, and P. Schneider-Kamp. Proving and disproving termination of higher-order functions. In *Proceedings of 5th International Symposium on Frontiers of Combining Systems (FroCoS '05)*, Volume 3717 of Lecture Notes in Artificial Intelligence, pages 216–231, 2005.

[29] J. Giesl, P. Schneider-Kamp, and R. Thiemann. AProVE 1.2: Automatic termination proofs in the dependency pair framework. In *Proceedings of 3rd International Joint Conference on Automated Reasoning (IJCAR '06)*, Volume 4130 of Lecture Notes in Artificial Intelligence, pages 281–286, 2006.

[30] J. Giesl, S. Swiderski, P. Schneider-Kamp, and R. Thiemann. Automated termination analysis for Haskell: From term rewriting to programming languages. In *Proceedings of 17th International Conference on Rewriting Techniques and Applications (RTA '06)*, Volume 4098 of Lecture Notes in Computer Science, pages 297–312, 2006.

[31] J. Giesl, R. Thiemann, P. Schneider-Kamp, and S. Falke. Mechanizing and improving dependency pairs. *Journal of Automated Reasoning*, 37(3):155–203, 2006.

[32] J. Giesl, M. Raffelsieper, P. Schneider-Kamp, S. Swiderski, and R. Thiemann. Automated termination proofs for Haskell by term rewriting. *ACM Transactions on Programming Languages and Systems*, 33(7), 2011.

[33] I. Gnaedig and P. Lescanne. Proving termination of associative commutative rewriting systems by rewriting. In *Proceedings of 8th International Conference on Automated Deduction (CADE '86)*, Volume 230 of Lecture Notes in Computer Science, pages 52–61, 1986.

[34] N. Hirokawa and A. Middeldorp. Dependency pairs revisited. In *Proceedings of 15th International Conference on Rewriting Techniques and Applications (RTA '04)*, Volume 3091 of Lecture Notes in Computer Science, pages 249–268, 2004.

[35] N. Hirokawa and A. Middeldorp. Polynomial interpretations with negative coefficients. In *Proceedings of 7th International Conference on Artificial Intelligence and Symbolic Mathematical Computation (AISC '04)*, Volume 3249 of Lecture Notes in Artificial Intelligence, pages 185–198, 2004.

[36] N. Hirokawa and A. Middeldorp. Automating the dependency pair method. *Information and Computation*, 199(1,2):172–199, 2005.

[37] N. Hirokawa, A. Middeldorp, and H. Zankl. Uncurrying for termination and complexity. *Journal of Automated Reasoning*, 50(3):279–315, 2013.

[38] D. Hofbauer and J. Waldmann. Termination of string rewriting with matrix interpretations. In *Proceedings of 16th International Conference on Rewriting Techniques and Applications (RTA '06)*, Volume 4098 of Lecture Notes in Computer Science, pages 328–342, 2006.

[39] H. Hong and D. Jakuš. Testing positiveness of polynomials. *Journal of Automated Reasoning*, 21:23–38, 1998.

[40] S. Kamin and J.-J. Lévy. Two generalizations of the recursive path ordering, 1980. Unpublished note.

[41] D. Kapur and G. Sivakumar. Proving associative-communicative termination using RPO-compatible orderings. In *Automated Deduction in Classical and Non-Classical Logics, Selected Papers*, Volume 1761 of Lecture Notes in Computer Science, pages 39–61, 1998.

[42] D. Kapur, G. Sivakumar, and H. Zhang. A new method for proving termination of AC-rewrite systems. In *Proceedings of 10th Foundations of Software Technology and Theoretical Computer Science (FSTTCS '90)*, Volume 472 of Lecture Notes in Computer Science, pages 133–148, 1990.

[43] D.E. Knuth and P. Bendix. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, New York, 1970.

[44] A. Koprowski and J. Waldmann. Max/plus tree automata for termination of term rewriting. *Acta Cybernetica*, 19(2):357–392, 2009.

[45] K. Korovin and A. Voronkov. Orienting rewrite rules with the Knuth-Bendix order. *Information and Computation*, 183(2):165–186, 2003.

[46] K. Korovin and A. Voronkov. An AC-compatible Knuth-Bendix order. In *Proceedings of 19th International Conference on Automated Deduction (CADE '03)*, Volume 2741 of Lecture Notes in Artificial Intelligence, pages 47–59, 2003.

[47] M. Korp, C. Sternagel, H. Zankl, and A. Middeldorp. Tyrolean Termination Tool 2. In *Proceedings of 20th International Conference on Rewriting*

Techniques and Applications (RTA '09), Volume 5595 of Lecture Notes in Computer Science, pages 295–304, 2009.

[48] L. Kovács, G. Moser, and A. Voronkov. On transfinite Knuth-Bendix orders. In *Proceedings of 23rd International Conference on Automated Deduction (CADE '11)*, Volume 6803 of Lecture Notes in Artificial Intelligence, pages 384–399, 2011.

[49] J. Kruskal. Well-quasi-ordering, the tree theorem, and Vazsonyia's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960.

[50] K. Kusakari. *AC-Termination and Dependency Pairs of Term Rewriting Systems*. PhD thesis, JAIST, 2000.

[51] K. Kusakari and Y. Toyama. On proving AC-termination by AC-dependency pairs. *IEICE Transactions on Information and Systems*, E84-D(5):439–447, 2001.

[52] K. Kusakari, M. Nakamura, and Y. Toyama. Argument filtering transformation. In *Proceedings of 1st International Symposium on Principles and Practice of Declarative Programming (PPDP '99)*, Volume 1702 of Lecture Notes in Computer Science, pages 47–61, 1999.

[53] D. Lankford. Canonical algebraic simplification in computational logic. Technical Report ATP-25, University of Texas, 1975.

[54] D. Lankford. Some approaches to equality for computational logic: A survey and assessment. Technical Report ATP-36, University of Texas, 1977.

[55] D. Lankford. On proving term rewrite systems are Noetherian. Technical Report MTP-3, Louisiana Technical University, 1979.

[56] P. Lescanne. Computer experiments with the REVE term rewriting system generator. In *Proceedings of 10th ACM SIGACT-SIGPLAN symposium on Principles of programming languages (POPL '83)*, pages 99–108, 1983.

[57] B. Löchner. Things to know when implementing KBO. *Journal of Automated Reasoning*, 36(4):289–310, 2006.

[58] M. Ludwig and U. Waldmann. An extension of the Knuth-Bendix ordering with LPO-like properties. In *Proceedings of 14th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '07)*, Volume 4790 of Lecture Notes in Artificial Intelligence, pages 348–362, 2007.

[59] Z. Manna and S. Ness. On the termination of Markov algorithms. In *Proceedings of the 3rd Hawaii International Conference on System Science*, pages 789–792, 1970.

[60] C. Marché and X. Urbain. Modular and incremental proofs of AC-termination. *Journal of Symbolic Computation*, 38(1):873–897, 2004.

[61] U. Martin. How to choose the weights in the Knuth Bendix ordering. In *Proceedings of 2nd International Conference on Rewriting Techniques and Applications (RTA '87)*, pages 42–53, 1987.

[62] A. Middeldorp and H. Zantema. Simple termination of rewrite systems. *Theoretical Computer Science*, 175(1):127–158, 1997.

[63] C. Otto, M. Brockschmidt, C. von Essen, and J. Giesl. Automated termination analysis of java bytecode by term rewriting. In *Proceedings of 21st International Conference on Rewriting Techniques and Applications (RTA '10)*, Volume 6 of Leibniz International Proceedings in Informatics, pages 259–276, 2010.

[64] U.S. Reddy. Term rewriting induction. In *Proceedings of 10th International Conference on Automated Deduction (CADE '90)*, Volume 449 of Lecture Notes in Computer Science, pages 162–177, 1990.

[65] A. Rubio. A fully syntactic AC-RPO. *Information and Computation*, 178 (2):515–533, 2002.

[66] A. Rubio and R. Nieuwenhuis. A total AC-compatible ordering based on RPO. *Theoretical Computer Science*, 142(2):209–227, 1995.

[67] F. Schernhammer and B. Gramlich. VMTL – a modular termination laboratory. In *Proceedings of 20th International Conference on Rewriting Techniques and Applications (RTA '09)*, Volume 5595 of Lecture Notes in Computer Science, pages 285–294, 2009.

[68] P. Schneider-Kamp, J. Giesl, A. Serebrenik, and R. Thiemann. Automated termination analysis for logic programs by term rewriting. In *Proceedings of 16th International Symposium on Logic-Based Program Synthesis and Transformation (LOPSTR '06)*, Volume 4407 of Lecture Notes in Computer Science, pages 177–193, 2007.

[69] J. Steinbach. Extensions and comparison of simplification orders. In *Proceedings of 3rd International Conference on Rewriting Techniques and Applications (RTA '89)*, Volume 355 of Lecture Notes in Computer Science, pages 434–448, 1989.

[70] J. Steinbach. AC-termination of rewrite systems: A modified Knuth-Bendix ordering. In *Proceedings of 2nd International Conference on Algebraic and Logic Programming (ALP '90)*, Volume 463 of Lecture Notes in Computer Science, pages 372–386, 1990.

[71] C. Sternagel and R. Thiemann. Generalized and formalized uncurrying. In *Proceedings of 8th International Symposium on Frontiers of Combining Systems (FroCoS '11)*, Volume 6989 of Lecture Notes in Artificial Intelligence, pages 243–258, 2011.

[72] C. Sternagel and R. Thiemann. Formalizing Knuth-Bendix orders and Knuth-Bendix completion. In *Proceedings of 24th International Conference on Rewriting Techniques and Applications (RTA '13)*, Volume 21 of Leibniz International Proceedings in Informatics, pages 287–302, 2013.

[73] T. Sternagel and H. Zankl. KBCV – knuth-bendix completion visualizer. In *Proceedings of 6th International Joint Conference on Automated Reasoning (IJCAR '12)*, Volume 7364 of Lecture Notes in Computer Science, pages 530–536, 2012.

[74] TeReSe. *Term Rewriting Systems*. Volume 55 of Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 2003.

[75] TermComp. The termination competition, 2013. URL `http://termcomp.uibk.ac.at/termcomp/`.

[76] R. Thiemann, G. Allais, and J. Nagele. On the formalization of termination techniques based on multiset orderings. In *Proceedings of 23rd International Conference on Rewriting Techniques and Applications (RTA '12)*, Volume 15 of Leibniz International Proceedings in Informatics, pages 339–354, 2012.

[77] TPDB. The termination problem data base, version 8.0.6, 2013. URL `http://termination-portal.org/wiki/TPDB`.

[78] I. Wehrman, A. Stump, and E. Westbrook. Slothrop: Knuth-Bendix completion with a modern termination checker. In *Proceedings of 17th International Conference on Rewriting Techniques and Applications (RTA '06)*, Volume 4098 of Lecture Notes in Computer Science, pages 287–296, 2006.

[79] S. Winkler. *Termination Tools in Automated Reasoning.* PhD thesis, University of Innsbruck, 2013.

[80] S. Winkler and A. Middeldorp. Termination tools in ordered completion. In *Proceedings of 5th International Joint Conference on Automated Reasoning (IJCAR '10)*, Volume 6173 of Lecture Notes in Artificial Intelligence, pages 518–532, 2010.

[81] S. Winkler, H. Zankl, and A. Middeldorp. Ordinals and Knuth-Bendix orders. In *Proceedings of 18th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '12)*, Volume 7180 of LNCS Advanced Research in Computing and Software Science, pages 420–434, 2012.

[82] S. Winkler, H. Sato, A. Middeldorp, and M. Kurihara. Multi-completion with termination tools. *Journal of Automated Reasoning*, 50(3):317–354, 2013.

[83] S. Winkler, H. Zankl, and A. Middeldorp. Beyond Peano arithmetic – automatically proving termination of the Goodstein sequence. In *Proceedings of 24th International Conference on Rewriting Techniques and Applications (RTA '13)*, Volume 21 of Leibniz International Proceedings in Informatics, pages 335–351, 2013.

[84] H. Xi. Towards automated termination proofs through "freezing". In *Proceedings of 9th International Conference on Rewriting Techniques and Applications (RTA '98)*, Volume 1379 of Lecture Notes in Computer Science, pages 271–285, 1998.

[85] A. Yamada, K. Kusakari, and T. Sakabe. Unifying the Knuth-Bendix, recursive path and polynomial orders. In *Proceedings of 15th International Symposium on Principles and Practice of Declarative Programming (PPDP '13)*, pages 181–192, 2013.

[86] A. Yamada, K. Kusakari, and T. Sakabe. Partial status for KBO. In *Proceedings of 13th International Workshop on Termination (WST '13)*, pages 74–78, 2013.

[87] A. Yamada, K. Kusakari, and T. Sakabe. A unified order for termination proving. *Science of Computer Programming*, 2014. To apear, available as CoRR abs/1404.6245.

[88] A. Yamada, K. Kusakari, and T. Sakabe. Nagoya Termination Tool. In *Proceedings of Joint 25th International Conference on Rewriting Techniques and Applications and 12th International Conference on Typed Lambda Calculi and Applications (RTA-TLCA '14)*, Volume 8560 of Lecture Notes in Computer Science, pages 466–475, 2014. To appear, full version appeared as CoRR abs/1404.6626.

[89] A. Yamada, S. Winkler, N. Hirokawa, and A. Middeldorp. AC-KBO revisited. In *Proceedings of 12th International Symposium on Functional and Logic Programming (FLOPS '14)*, Volume 8475 of Lecture Notes in Computer Science, pages 319–335, 2014. Full version appeared as CoRR abs/1403.0406.

[90] H. Zankl and A. Middeldorp. Satisfiability of non-linear (ir)rational arithmetic. In *Proceedings of 16th International Conference on Logic for Programming, Artificial Intelligence and Reasoning (LPAR '10)*, Volume 6355 of Lecture Notes in Artificial Intelligence, pages 481–500, 2010.

[91] H. Zankl, N. Hirokawa, and A. Middeldorp. KBO orientability. *Journal of Automated Reasoning*, 43(2):173–201, 2009.

[92] H. Zantema. Termination of term rewriting: interpretation and type elimination. *Journal of Symbolic Computation*, 17(1):23–50, 1994.

[93] H. Zantema. The termination hierarchy for term rewriting. *Applicable Algebra in Engineering, Communication and Computing*, 12:3–19, 2001.