# KNUTH–BENDIX ORDERS IN AUTOMATED DEDUCTION AND TERM REWRITING

A thesis submitted to the University of Manchester
for the degree of Doctor of Philosophy
in the Faculty of Science and Engineering

2003

By
Konstantin Korovin
Department of Computer Science

# Contents

# Abstract

Ordering restrictions play a crucial role in automated deduction. In particular, orders are used extensively for pruning search space in automated theorem provers and for rewriting-based reasoning and computation. There are two classes of orders that are widely used in automated deduction: Knuth-Bendix orders and various versions of recursive path orders. Despite the fact that Knuth-Bendix orders were discovered earlier than recursive path orders, and since then have been used in many state-of-the-art automated theorem provers; the decidability and complexity of many important problems related to these orders remained open. In this thesis we try to close this gap and provide various decidability and complexity results for a number of important decision problems related to Knuth-Bendix orders. We prove the decidability and NP-completeness of the problem of solving Knuth-Bendix ordering constraints. In the case of constraints consisting of single inequalities we present a polynomial-time algorithm. We also prove the decidability of the problem of solving general first-order Knuth-Bendix ordering constraints over unary signatures. Another problem we study is the orientability problem by Knuth-Bendix orders. We present a polynomial-time algorithm for orientability of systems consisting of term rewrite rules and equalities by Knuth-Bendix orders, and prove that this problem is P-complete. Finally, we show that it is possible to extend Knuth-Bendix orders to AC-compatible orders preserving attractive properties of Knuth-Bendix orders.

# Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

# Copyright

# Acknowledgments

I am very grateful to my supervisor Professor Andrei Voronkov who introduced me to the area of automated deduction and with whom it has been a great pleasure to work. My colleagues at the Department of Computer Science has been always supportive and open for discussions, which has helped me much in my research.

I thank Professor Harald Ganzinger who gave me an opportunity to visit Max-Planck Institute für Informatik where I have had an excellent research experience.

I also acknowledge ORS Awards Scheme and Department of Computer Science, University of Manchester for supporting me financially.

# Chapter 1

# Introduction

Automated deduction is an important branch of Computer Science, which has applications in various areas including specification and verification of software and hardware, synthesis of safe programs, database systems, computer algebra and others. One of the most popular methods used in automated deduction is resolution-based theorem proving. It turns out that this method is powerful enough for many applications, yet it can be implemented efficiently. Resolution-based theorem proving was introduced by Robinson in his seminal paper [Robinson 1965]. Because of its practical importance, a huge amount of research has been devoted to theoretical improvements of this method, likewise to efficient implementation issues. Introduction of ordering restrictions has been one of the main breakthroughs in resolution-based theorem proving and in equational reasoning. In this work we are mainly focused on theoretical problems related to ordering restrictions that can help to improve performance of resolution-based theorem provers. Major research directions involving orders for automated deduction include

- solving ordering constraints,
- orientability problems,
- studying orders compatible with various equational theories, and
- efficient ordering algorithms.

There are two classes of orders that are widely used in automated deduction: Knuth-Bendix orders [Knuth and Bendix 1970] and various versions of recursive path orders [Dershowitz 1982]. Both Knuth-Bendix orders and recursive path orders are used in most of the state-of-the-art theorem provers, for example Vampire [Riazanov and Voronkov 1999], E [Schulz 1999], Waldmeister [Hillenbrand, Buch, Vogt and Löchner 1997] and SPASS [Weidenbach 2001]. During the last two decades recursive path orders have been intensively studied and many important results have been obtained by various researchers. Despite the fact that Knuth-Bendix orders were discovered earlier than recursive path orders and since then used in most of automated theorem provers, almost nothing had been known about properties of these orders.

In this work we try to close this gap and provide various decidability and complexity results for a number of important problems related to Knuth-Bendix orders. Let us draw some connections between known results in the area and results of this thesis. More details about applications of orders in automated deduction can be found in Chapter 2.

**Constraint solving.** Using solvability of ordering constraints we can dramatically reduce the number of redundant inferences in a resolution-based prover. As a consequence, the problem of solving ordering constraints for the known simplification orders is one of the important problems in the area. There exists extensive literature on solving recursive path ordering constraints: [Jouannaud and Okada 1991, Comon 1990, Nieuwenhuis 1993, Nieuwenhuis and Rivero 1999, Narendran, Rusinowitch and Verma 1998, Narendran and Rusinowitch 2000], but until recently no algorithms for solving Knuth-Bendix ordering constraints were known. We show

- The decidability and NP–completeness of the problem of solving Knuth-Bendix ordering constraints (see Chapter 4).
- The polynomial–time computability of the problem of solving Knuth-Bendix ordering constraints consisting of single inequalities (see Chapter 6).
- The decidability of first-order Knuth-Bendix ordering constraints over unary signatures (see Chapter 5).

These results are reported in [Korovin and Voronkov 2000, Korovin and Voronkov 2001*a*, Korovin and Voronkov 2001*b*, Korovin and Voronkov 2002, Korovin and Voronkov 2003*b*].

**Orientability.** Usually an order is an important parameter of a deduction system that can be chosen according to the problem to solve. The choice of an order is especially important for problems containing equality and problems related to term rewrite systems and can be restated as a problem of orientability of equational (term rewrite) systems. In some cases the user can define manually the order that the system should use on a given problem. Of course it would be desirable to automate the process of choosing an appropriate order. In general this problem is bound to be computationally difficult, and hence we can try to solve this problem for some known classes of orders. The orientability problem for recursive path orders has been studied and shown to be NP–hard [Krishnamoorthy and Narendran 1985, Lescanne 1984]. We study orientability problem for Knuth-Bendix orders and show the following.

- The problem of the existence of a Knuth-Bendix order which orients a given system of equalities and term rewrite rules can be solved in the time polynomial in the size of the system. Moreover, if the system of equalities and rewrite rules is orientable by a Knuth-Bendix order, we can find such an instance in polynomial time (see Chapters 6,7).
- The problem of orientability of systems of equalities and rewrite rules by a Knuth-Bendix order is P-complete. Moreover, it is P-hard even for systems consisting only of term rewrite rules or only of equalities (see Chapters 6,7).

These results are reported in [Korovin and Voronkov 2001*b*, Korovin and Voronkov 2003*c*, Korovin and Voronkov 2003*d*]. Let us note that an algorithm for orientability of term rewriting systems by a weaker version of Knuth-Bendix order has been presented in [Martin 1987, Dick, Kalmus and Martin 1990].

**Orders compatible with associativity–commutativity.** Among various equational theories, theories axiomatized by the axioms of associativity and commutativity, so-called AC-theories, play a special role. Such theories very often occur in applications and require special treatment in automated systems. In such systems AC-compatible simplification orders is a crucial ingredient. Importance of AC-compatible simplification orders triggered a huge amount of research aimed to design such orders: [Dershowitz, Hsiang, Josephson and Plaisted 1983, Bachmair and Plaisted 1985, Gnaedig and Lescanne 1986, Cherifa and Lescanne 1987, Kapur, Sivakumar and Zhang 1990, Narendran and Rusinowitch 1991, Bachmair

1992, Rubio and Nieuwenhuis 1993, Kapur, Sivakumar and Zhang 1995, Marché 1995, Kapur and Sivakumar 1997, Kapur and Sivakumar 1998, Rubio 1999, Rubio 2002]. Usually, AC-compatible simplification orders are designed from known simplification orders. Recently, a lot of work has been done to modify recursive path orders to obtain AC-compatible simplification orders AC-total on ground terms [Kapur et al. 1990, Rubio and Nieuwenhuis 1993, Kapur et al. 1995, Kapur and Sivakumar 1997, Kapur and Sivakumar 1998, Rubio 1999, Rubio 2002]. Although Knuth-Bendix orders are widely used in automated deduction, to our knowledge no AC-compatible simplification variant of Knuth-Bendix orders have been known. In Chapter 8 we define a family of AC-compatible Knuth-Bendix orders. These orders enjoy attractive features of the standard Knuth-Bendix orders, such as

- a polynomial-time algorithm for term comparison, and
- computationally efficient approximations.

These results are reported in [Korovin and Voronkov 2003*a*].

# Chapter 2

# Motivation

In this chapter we briefly introduce resolution and paramodulation calculi. The main goal of our presentation is to illustrate how results of this thesis can be applied in automated deduction. Thus, all results in this chapter are well-known and the reader wishing to study this subject in detail can find a comprehensive treatment of these topics in e.g. [Bachmair and Ganzinger 2001, Baader and Nipkow 1998, Nieuwenhuis and Rubio 2001].

This chapter is organized as follows. In Section 2.2 we introduce a simple version of resolution-based inference system and discuss the efficiency problems. In Section 2.3 we show how these problems can be tackled with the help of simple constraints. In Section 2.4 we extend resolution into resolution with constrained clauses. In Section 2.5 we introduce the subsumption rule and show how first-order constraints can be used. In Section 2.6 we introduce equational reasoning and term rewriting, and discuss the role of orientability. In Section 2.7 we show how the resolution system can be extended with equality and the use of constraint solving and orientability for this system. Finally, in Section 2.8 we show how to integrate nonorientable equations like commutativity into term rewriting and paramodulation calculi with the help of $E$-compatible simplification orders.

## 2.1 Introduction

In practical applications we can specify properties of systems such as programs or hardware devices using first-order formulas. Usually we want to be sure that this specification satisfies some required properties. Often problems of this kind can be reformulated as the validity problem for first-order formulas. In order to prove validity of first-order formulas various deduction calculi have been devised. Roughly speaking, deduction calculi allow us to prove validity of formulas using simple transformations (derivations). Two main properties of a deduction calculus are soundness and completeness. Soundness ensures that we deduce only valid formulas and completeness guarantees that if a given formula is valid then we can prove it in a finite number of steps.

Another important property of a deduction calculus is that it can be implemented efficiently. This is one of the main concerns in the area of automated deduction. It turns out that it is a very difficult task to devise an efficient calculus for first-order logic. One of the most successful attempts is the resolution calculus introduced by Robinson [1965]. The resolution calculus and its refinements form a basis for most of the contemporary theorem provers for first-order logic. Let us briefly describe this calculus.

## 2.2 Resolution-based theorem proving

We assume that the reader is familiar with the syntax and semantics of first-order logic. We consider formulas over a finite language consisting of predicate and function symbols and we assume that the language is arbitrary but fixed. Also w.l.o.g. we assume that our language contains at least one constant.

Let us sketch how the resolution calculus can be applied to prove validity of first-order formulas. To prove the validity of a first-order formula we prove the unsatisfiability of its negation. To prove the unsatisfiability of a formula we first eliminate all existential quantifiers, by a transformation preserving satisfiability/unsatisfiability of this formula (for efficient algorithms for such transformations we refer to [Baaz, Egly and Leitsch 2001, Nonnengart and Weidenbach 2001]). Now we can restrict ourself to universally quantified formulas. The key theorem, which is used to prove completeness of resolution calculi, is Herbrand's theorem which states the following. Consider a formula $\phi = \forall \bar{x} \psi(x)$ where $\psi(\bar{x})$ is a quantifier–free formula. Then, $\phi$ is unsatisfiable if and only if there exists a finite number of tuples of terms $\bar{t}_1, \ldots, \bar{t}_n$ without variables such that the formula $\psi(\bar{t}_1) \wedge \ldots \wedge \psi(\bar{t}_n)$ is unsatisfiable. Let us note that this theorem gives us a semi–decision procedure for proving validity of first-order formulas, since we can check effectively satisfiability of variable–free formulas. Of course, such a procedure would be highly inefficient in practice, that is the reason why the resolution calculus was devised.

Now we are ready to introduce the resolution calculus. The resolution calculus involves formulas of a special kind, so-called clauses. A *clause* is a disjunction of literals, where a *literal* is either an atomic or a negated atomic formula. Initially we have a set of clauses which are implicitly universally quantified. The goal is to prove that this set is unsatisfiable, or in other words to deduce the empty clause. The inference system consists of two rules: the resolution rule and the factoring rule.

**Resolution:** $$\frac{A \vee C \quad \neg B \vee D}{(C \vee D)\theta}$$

where $\theta$ is the most general unifier of the atoms $A$ and $B$.

**Factoring:** $$\frac{A \vee B \vee C}{(B \vee C)\theta}$$

where $\theta$ is the most general unifier of the atoms $A$ and $B$.

This inference system was proved to be refutation complete, i.e., if we have an unsatisfiable set of clauses, then there is a proof of the empty clause using these inference rules. Usually the proof search is implemented via a saturation process, i.e., exhaustively application of inferences to the previously derived clauses. There are three possible outcomes of a saturation process. We derive the empty clause which means that the initial set of clauses is unsatisfiable. Or, the procedure stops without deducing the empty clause which means that the initial set of clauses is satisfiable. The third possibility is that the procedure does not terminate which means that the initial set of clauses is satisfiable. Only the first two outcomes are useful for applications. Consequently we want to restrict nontermination of the procedure without compromising completeness. Although we can not avoid nontermination of resolution process on all problems, due to undecidability of first-order logic, we can narrow the class of such problems.

Let us consider the following simple example.

Example 2.2.1 Consider the following set of clauses $S = \{B(x) \vee A(f(x)), \neg A(x) \vee A(f(x))\}$. It is easy to see that $S$ is satisfiable, nevertheless the resolution process does not terminate.

$$\cfrac{\cfrac{\cfrac{B(x) \vee A(f(x)) \quad \neg A(x) \vee A(f(x))}{B(x) \vee A(f(f(x)))} \quad \neg A(x) \vee A(f(x))}{B(x) \vee A(f(f(f(x))))} \quad \neg A(x) \vee A(f(x))}{\cdots}$$

It turns out that even for unsatisfiable sets of clauses, the straightforward approach of applying inference rules is very inefficient in the sense that we generate a huge number of unnecessary inferences. Therefore, one of the main problems in the area is how to restrict applicability of the inference rules while preserving the completeness of the inference system. One of the most prominent approaches to this problem is based on various ordering restrictions on applicability of inference rules. Ordering restrictions and related problems will be the main topic for the rest of this chapter.

## 2.3 Resolution and constraints

Ordering refinements were introduced into resolution in [Slagle 1967], who attributes the idea to [Reynolds 1965]. In [Slagle 1967] orders on literals in the clause were used to restrict applicability of resolution and factoring rules. This idea turned out to be very productive (see e.g. [Bachmair and Ganzinger 2001] for a comprehensive recent survey). In particular, if we consider a simplification order $\succ$ (see Definition 3.3.1) on the set of ground atoms, then the following resolution system with ordering restrictions is complete. (To simplify the presentation we omit restrictions based on selection functions and refer to [Bachmair and Ganzinger 2001] for the general case.)

**Resolution:** $\dfrac{A \vee C \quad \neg B \vee D}{(C \vee D)\theta}$

where $\theta$ is the most general unifier of the atoms $A$ and $B$.

**Restriction of applicability:** For every atom $C'$ in $C$ there exists a ground substitution $\gamma$ such that $A\theta\gamma \succ C'\theta\gamma$. In other words, we apply this inference rule only if the ordering constraint $A\theta(\bar{x}) \succ C'\theta(\bar{x})$ is satisfiable. Likewise, for every atom $D'$ in $D$ there exists a ground substitution $\sigma$ such that $B\theta\sigma \succeq D'\theta\sigma$. So, in addition we require that the ordering constraint $B\theta(\bar{x}) \succeq D'\theta(\bar{x})$ is satisfiable.

**Factoring:** $\dfrac{A \vee B \vee C}{(B \vee C)\theta}$

where $\theta$ is the most general unifier of the atoms $A$ and $B$.

**Restriction of applicability:** For every atom $C'$ in $C$ there exists a ground substitution $\gamma$ such that $A\theta\gamma \succeq C'\theta\gamma$. In other words, we apply this inference rule only if the ordering constraint $A\theta(\bar{x}) \succeq C'\theta(\bar{x})$ is satisfiable.

These ordering restrictions are powerful but to use them we need algorithms for solving ordering constraints, (see Chapter 3 for the definition of constraints). There are two classes of orders extensively used in automated deduction, namely Knuth-Bendix orders and recursive path orders. The decidability of recursive path ordering constraints is shown in [Comon 1990] and complexity results are given in [Jouannaud and Okada 1991, Nieuwenhuis 1993, Comon and Treinen 1994]. In Chapter 4 we prove the decidability of Knuth-Bendix ordering constraints and show that this problem is NP-complete for conjunctive constraints (as corollary it is NP-complete for quantifier free constraints). It is interesting to note that for Knuth-Bendix constraints consisting of a single inequality, as used above in ordered resolution, there is an efficient polynomial-time algorithm solving them, presented in Chapter 6. This is in contrast with recursive path orders, for which it is shown that the problem of solving constraints consisting of a single inequality is NP-complete [Comon and Treinen 1994].

Let us reconsider Example 2.2.1. Now we apply ordered resolution instead of unrestricted resolution. For a suitable order we can have that the constraint $A(f(x)) \succ B(x)$, is unsatisfiable and therefore the procedure stops returning the answer "satisfiable", in contrast to the unrestricted resolution. (An example of such an order is a Knuth-Bendix order $\succ$ with parameters $\{|B| = 3, |A| = 1, |f| = 1; B \gg A \gg f\}$ see Definition 3.3.8 of Knuth–Bendix orders.)

Here we also can notice that in addition to the constraint satisfaction problem, there is a problem of choosing an appropriate order to minimize the number of applicable rules. This problem is related to the orientability problem, which is shown to be decidable in polynomial time for Knuth-Bendix orders see Chapters 6,7.

It turns out that it is possible to restrict resolution even further by introducing constrained clauses, which will be discussed in the next section.

## 2.4 Inherited constraints

In order to restrict resolution further, instead of ordinary clauses we consider *constrained clauses* which are of the form $C(\bar{x}) \mid \phi(\bar{x})$, where $C(\bar{x})$ is a clause and $\phi(\bar{x})$ is an ordering constraint. Usually a clause is viewed as a representation of all its ground instances $C(\bar{x})\sigma$, then a constrained clause $C(\bar{x}) \mid \phi(\bar{x})$ can be viewed as a representation of all ground $C(\bar{x})\sigma$ such that the constraint $\phi(x)\sigma$

is valid. The main benefit of using constrained clauses is that we can inherit constraints along the derivation. Now the resolution rule can be replaced with the following rule.

**Resolution with inherited constraints:** $$\frac{C \vee A \mid T \quad \neg B \vee D \mid T'}{(C \vee D)\theta \mid (OC \wedge T \wedge T')\theta}$$

where $\theta$ is the most general unifier of the atoms $A$ and $B$ and $OC$ is the ordering constraint imposed by this inference. Now a clause $C(\bar{x}) \mid \phi(\bar{x})$ is redundant if constraint $\phi(\bar{x})$ is unsatisfiable.

Various types of constraint clauses are introduced and completeness results are proved in [Huet 1972, Bürkert 1990, Kirchner, Kirchner and Rusinowitch 1990, Nieuwenhuis and Rubio 1992, Nieuwenhuis and Rubio 1995]. Again, in order to gain from constrained clauses, we need algorithms for checking solvability of ordering constraints (see Chapter 4 for a nondeterministic polynomial time algorithm for this problem for Knuth-Bendix orders).

## 2.5 First-order constraints

In resolution-based theorem proving there are important simplifications which allow us to remove clauses from the search space. It turns out that in order to express applicability conditions for these simplifications, we need to consider constraints which involve first-order quantifiers. As an example we consider subsumption.

***Subsumption:*** ([Voronkov 2000]) We say that a constrained clause $C(\bar{x}) \mid \varphi(\bar{x})$ subsumes a constrained clause $D(\bar{x}) \mid \psi(\bar{x})$ if the following holds:

$$\forall x(\psi(x) \rightarrow \exists y(\varphi(y) \wedge C(y) \subseteq D(x))).$$

If the clause $D(\bar{x}) \mid \psi(\bar{x})$ is subsumed by a clause $C(\bar{x}) \mid \varphi(\bar{x})$ then it can be shown that the clause $D(\bar{x}) \mid \psi(\bar{x})$ is redundant and can be removed from the search space. In order to check whether one clause is subsumed by another we need to solve ordering constraints involving alternation of quantifiers. Unfortunately the first-order theory of recursive path orders is undecidable [Treinen 1990, Comon and Treinen 1997]. Recently, it was shown that in the case of the signatures consisting of unary function symbols and constants the first-order theory of recursive path orders is decidable [Narendran and

Rusinowitch 2000]. In Chapter 5 we show that the first-order theory of the Knuth–Bendix orders is decidable if we consider signatures consisting of unary function symbols and constants.

Another possible application of solvability of first-order ordering constraints is simplification of constraints. For example, consider a constrained clause $C(\bar{x}) \mid \varphi(\bar{x}, \bar{y})$. It might be the case that the variables $\bar{y}$ do not occur in the clause $C(\bar{x})$ and therefore we want to simplify the constraint $\varphi(\bar{x}, \bar{y})$ to a constraint $\varphi'(\bar{x})$ which does not contain variables from $\bar{y}$. From the decidability procedure for first-order Knuth–Bendix ordering constraints over unary signatures, we can see that there is a representation of constraints where such redundant variables can be eliminated.

## 2.6 Equational reasoning and term rewriting

Equational reasoning plays an important role in mathematics and computer science. Most problems occurring in practical applications involve reasoning with equality.

Formally, we are studying properties of structures defined using identities. Although the language is restrictive we still can define a lot of interesting and important classes of structures such as groups, rings, lattices, etc.. Let us consider axioms of group theory:

- associativity axiom: $(x \circ y) \circ z \simeq x \circ (y \circ z)$ ,
- left-unit axiom: $e \circ x \simeq x$,
- left-inverse axiom: $i(x) \circ x \simeq e$.

In many situations we are interested in the following question: given a set of axioms and an equality $t \simeq s$, is $t \simeq s$ valid in all structures satisfying these axioms (e.g. is $i(x \circ y) = i(y) \circ i(x)$ valid in all groups)? In other words, does the given equality logically follow from the axioms? One way to check it, is to transform terms $t$ and $s$ by replacing equal subterms using the axioms, and wait until $t$ will be syntactically equal to $s$. In fact, this method is sound and complete by Birkhoff's theorem (see e.g. [Baader and Nipkow 1998]), i.e., if the equality $t \simeq s$ follows from the axioms, then exhaustively applying transformations as above we will deduce syntactically identical terms in a finite number of steps.

Unfortunately this method has two major drawbacks. First, for a given set of axioms we cannot predict whether the algorithm terminates for every equality $t \simeq s$. More importantly, this algorithm is hopelessly inefficient. For example if we want to prove that $(x \circ i(y)) \circ y \simeq x$ follows from the axioms of group theory, then among other deduced equalities we obtain $\{(x \circ (e \circ i(y))) \circ y \simeq x, (x \circ i(y)) \circ (e \circ y) \simeq e \circ x, (x \circ i(e \circ y)) \circ (e \circ y) \simeq x) \ldots\}$ (using left-unit axiom), which have nothing to do with the actual proof.

The main approach to overcome these problems is as follows. We represent axioms as rewrite rules and apply them only in one direction. Now in place of axioms we have term rewrite rules. For example a possible term rewriting system for groups is as follows.

- associativity rule: $(x \circ y) \circ z \to x \circ (y \circ z)$ ,
- left-unit rule: $e \circ x \to x$,
- left-inverse rule: $i(x) \circ x \to e$.

The idea is to reduce a given term into a normal form using these rewrite rules. Then, if our term rewriting system satisfies certain properties, we can guarantee that this rewriting process will always terminate and produce a unique normal form for each term. As a consequence, the problem of checking whether a given equality follows from the axioms becomes simple: we produce normal forms of corresponding terms and check syntactic identity of normal forms. This approach, called term rewriting, was introduced in the seminal paper of Knuth and Bendix [1970] and has been intensively studied and developed during the last 30 years.

For example, consider again the equality $(x \circ i(y)) \circ y \simeq x$. The only applicable rule is the associativity rule which produces $x \circ (i(y) \circ y) \simeq x$, at the next step the only applicable rule is the left-inverse rule which produces $x \circ e \simeq x$, now the only applicable rule is the left-unit rule which proves the equality producing $x \simeq x$. Although the term rewriting system above is sound, it is incomplete, i.e., not all equalities which follow from the axioms of group theory can be proved by this term rewriting system. For example $x \circ i(x) \simeq e$ is a logical consequence of group theory, but cannot be proved by this term rewriting system (none of the rules is applicable). Therefore a natural question to ask is what are the properties of term rewriting systems which guarantee that the term rewriting system is complete, i.e., can prove all logical consequences? These properties are termination and confluence. Termination guarantees that there is no infinite sequences of rewritings and confluence guarantees that if we can rewrite a given term into two different ones then we can join these rewrites. If a term rewriting system is confluent and terminating, then every term can be rewritten into a unique normal form. As a consequence, every equality can be proved or disproved by rewriting. Therefore, termination is one of the crucial properties of term rewriting systems. Moreover, it turns out that confluence is decidable if our term rewriting system is terminating. In general, termination of rewriting systems is undecidable (see e.g. [Baader and Nipkow 1998]), but in many practical cases we can prove termination using orientability of term rewriting systems by reduction orders. In fact, if our term rewriting system can be oriented using a reduction order then it is terminating. Let us define the orientability problem. Let $\succ$ be any reduction order on ground terms and $l \to r$ be a rewrite rule. We say that $\succ$ *orients* $l \to r$, if for every ground instance $l' \to r'$ of $l \to r$ we have $l' \succ r'$. We say that $\succ$ *orients* a term rewriting system $R$ if it orients every rewrite rule in $R$.

***Orientability problem (TRS)*:** Given a term rewriting system $R$ check whether there exists a reduction order $\succ$ which orients $R$.

Knuth-Bendix orders and recursive path orders are two major classes of orders that can be used to show termination of term rewriting systems. For recursive path orders the orientability problem is computationally difficult, in particular it is NP-hard and co-NP-hard [Krishnamoorthy and Narendran 1985, Comon and Treinen 1994]. We show that for Knuth-Bendix orders the orientability problem can be solved in polynomial-time, in particular we show that this problem is P-complete (Chapter 6). Let us note that there are powerful extentions of termination analysis based on orientability, by considering dependency relation between term rewrite rules, focusing only on rules that can start a nonterminating sequence of rewrites (see [Arts and Giesl 2000]).

The term rewriting technique for equational reasoning can be integrated into resolution-based theorem proving as shown in the next section. Let us mention that term rewriting systems are already expressive enough to be used in verification (see e.g. [Arts and Giesl 2001, Hoe and Arvind 1999]) where certain specifications are represented as term rewriting systems. Again, termination plays a crucial role there.

## 2.7 Introducing equality into resolution

The equality predicate is used in many applications and consequently it is important to introduce it into resolution calculus. One way to do this is by introducing equality axioms. Indeed, the equality predicate $\simeq$ can be axiomatized by the following set of axioms

- reflexivity axiom: $x \simeq x$;
- symmetry axiom: $x \simeq y \supset y \simeq x$;
- transitivity axiom: $x \simeq y \wedge y \simeq z \supset x \simeq z$;
- function substitution axioms: $x_1 \simeq y_1 \wedge \ldots \wedge x_n \simeq y_n \supset f(x_1, \ldots, x_n) \simeq f(y_1, \ldots, y_n)$, for every function symbol $f$;
- predicate substitution axioms: $x_1 \simeq y_1 \wedge \ldots \wedge x_n \simeq y_n \wedge P(x_1, \ldots, x_n) \supset P(y_1, \ldots, y_n)$, for every predicate symbol $P$.

Suppose that we want to prove a theorem containing equality, then we can try to deduce it from the equality axioms above using resolution system. However, this would lead to a combinatorial explosion due to the universal applicability of the equality axioms.

In order to overcome these problems it has been suggested to build equality into resolution calculus via special rules. Such a rule, called *paramodulation*, was introduced in [Robinson and Wos 1969].

**Paramodulation:** $$\frac{s \simeq t \vee C_1 \quad L[s'] \vee C_2}{(L[t] \vee C_1 \vee C_2)\theta},$$

where $\theta$ is the most general unifier of $s'$ and $s$.

Robinson and Wos [1969] proved completeness of the system consisting of resolution, factoring and paramodulation in the presence of some addition axioms, called function reflexivity axioms. Later Brand [1975] proved that resolution, factoring and paramodulation is complete even without function reflexivity axioms. Nevertheless, unrestricted application of paramodulation is still very inefficient. Recent research has been aiming at various restrictions of applicability of paramodulation. One of the most prominent approaches is introducing ordering restrictions where we replace "bigger" terms by "smaller" ones, with respect to the given simplification order. The main idea goes back to term rewriting. Given a simplification order $\succ$, we can replace paramodulation with ordered paramodulation as follows.

**Ordered paramodulation:** $$\frac{s \simeq t \vee C_1 \quad L[s'] \vee C_2}{(L[t] \vee C_1 \vee C_2)\theta},$$

where $\theta$ is the most general unifier of $s'$ and $s$.

**Restriction of applicability:**

- $s'$ is not a variable;
- there exists a ground substitution $\gamma$ such that $s\theta\gamma \succ t\theta\gamma$.

Ordered paramodulation was introduced and studied in [Peterson 1983, Hsiang and Rusinowitch 1986] where completeness results are proved. Let us refer to [Nieuwenhuis and Rubio 2001] for a recent comprehensive survey of state-of-the-art refinements of the paramodulation calculus.

Here we can observe that if an order $\succ$ is such that for all ground substitutions $\sigma$ we have $s\sigma \succ t\sigma$ then we can apply paramodulation rule only when we replacing instances of $s$ by instances of $t$ but not vice versa. This is a desirable restriction of applicability. Now we are facing a problem of how to choose an order such that equalities occurring in the set of clauses would be oriented by this order. This is the orientability problem for sets of equalities, which can be stated as follows. We say that $\succ$ *orients an equality* $s \simeq t$, if it orients either the rewrite rule $s \rightarrow t$ or the rewrite rule $t \rightarrow s$. The *orientability problem for systems of equalities* is a problem of determining whether there exists a simplification order which orients a given system of equalities. A straightforward algorithm for checking orientability of systems of equalities would be to try all possible orientations of equalities and apply an orientability algorithm for term rewriting systems. Such an algorithm would require to test an exponential number of possible orientations of equalities. In Chapter 7 we show how to overcome this problem for Knuth-Bendix orders, presenting a polynomial time algorithm for checking orientability of systems of equalities. In some cases orientation of some subsystem of equalities is desirable to be fixed in advance. For example, if we know which orientation of the group theory axioms can lead to a convergent term rewriting system, we might require that this subsystem be oriented in this particular way. So the general statement of the orientability problem is as follows: given a system of equalities and term rewrite rules, determining whether there exists a simplification order which orients this system. In Chapter 7 we show that this problem can also be solved in polynomial time for Knuth-Bendix orders.

A further refinement of ordered paramodulation is *maximal paramodulation.*

**Maximal paramodulation:** $$\frac{s \simeq t \vee C_1 \quad L[s'] \vee C_2}{(L[t] \vee C_1 \vee C_2)\theta},$$

where $\theta$ is the most general unifier of $s'$ and $s$.

**Restriction of applicability:**

**1.** $s'$ is not a variable;

**2.** there exists a ground substitution $\gamma$ such that $s\theta\gamma \succ t\theta\gamma$;

**3.** $L[s']\theta$ is maximal w.r.t. $\succ$ in $(L[s'] \vee C_2)\theta$;

**4.** $(s \simeq t)\theta$ is maximal w.r.t. $\succ$ in $(s \simeq t \vee C_1)\theta$.

Similar to resolution we can inherit constraints along the derivations without loosing completeness (see [Nieuwenhuis and Rubio 1995]). This imposes stronger restrictions on applicability of rules. Thus, we can make use of both orientability and constraint solving algorithms.

The rules described so far were inference rules, so every application of such a rule produces a new clause, therefore enlarging the search space. For efficiency reasons, another type of rules, so-called simplification rules, are of great importance. Simplification rules allow us to replace clauses with "simplified" ones. One of the most popular simplification rules for equality reasoning is *demodulation.*

**Demodulation:** $$\frac{s \simeq t \quad \boxed{L[s'] \vee C}}{(L[t] \vee C)\theta},$$

where $s' = s\theta$.

**Applicability:** $s\theta\gamma \succ t\theta\gamma$ for every ground substitution $\gamma$.

After application of the demodulation the clause in the frame will be removed from the search space. As a consequence we want to orient equations in order to simplify clauses. For this, we can again employ an orientability algorithm.

## 2.8 Building in equational theories

One disappointing feature of the term rewriting approach is that some important axioms like commutativity can be oriented by no simplification ordering. To cope with this problem rewriting modulo theories has been devised, where we rewrite equivalence classes generated by equational theory rather than individual terms. A general approach is to partition a given set of equational axioms into a set of rewrite rules $R$ which induces a rewrite relation on terms $\to_R$, and a set of equations $E$ which induces an equivalence relation on terms $=_E$. We say that a term $t$ $R/E$-rewrites in one step to a term $t'$ (and denote this by $t \to_{R/E} t'$) if there exists a term $s$ $E$-equivalent to $t$ and a term $s'$ $E$-equivalent to $t'$ such that $t$ rewrites to $t'$ by $\to_R$. In other words $t \to_{R/E} t'$ if $t =_E s[l\sigma]$ and $t' =_E s[r\sigma]$ for a term $s$ and a rewrite rule $l \to r$ in $R$. We say that a term $t$ is in a normal form if we cannot $R/E$-rewrite it. Now we can try to decide equational consequences of the the given set of axioms by normalizing terms w.r.t. $R/E$-rewriting and check whether the obtained normal forms are equivalent modulo $E$. For this, similar to the ordinary rewriting, $R/E$-rewriting has to be terminating and every two terms equal w.r.t. our axioms should rewrite to the same normal form. We can prove termination of $R/E$-rewriting using simplification orders which satisfy an additional property, called $E$-compatibility. We say that an order $\succ$ is *$E$-compatible* if it satisfies the following property: if $s \succ t$, $s =_E s'$ and $t =_E t'$, then $s' \succ t'$. The order $\succ$ is called *$E$-total*, if for all ground terms $s, t$, if $s \neq_E t$, then either $s \succ t$ or $t \succ s$. Designing $E$-compatible simplification orders has been an active research area.

Among various equational theories, theories axiomatized by the axioms of associativity and commutativity, so-called AC-theories, play a special role. Such theories very often occur in applications and require special treatment in automated systems. AC-reasoning based on AC-rewriting has been integrated into paramodulation framework in [Rusinowitch and Vigneron 1995, Nieuwenhuis and Rubio 1997]. A crucial ingredient in these approaches is an AC-compatible AC-total simplification order. Existence of an AC-compatible AC-total simplification order has been an open problem for many years and was finally solved by Narendran and Rusinowitch [1991] who applied this order to show that any ground AC-theory can be represented as a finite convergent rewriting system. Unfortunately this order was defined only for ground terms which restricts its applicability.

Recently there has been a huge amount of research devoted to designing AC-compatible AC-total simplification orders, mainly by modifying recursive path orders. Since Knuth-Bendix orders are widely used in automated deduction, it is important to find AC-compatible variants of it. In Chapter 8 we present a family of AC-compatible Knuth-Bendix orders. These orders enjoy attractive features of the standard Knuth-Bendix orders, such as polynomial-time algorithm for term comparison and computationally efficient approximations based on weight comparisons.

# Chapter 3

# Ordering restrictions: preliminaries

In this chapter we introduce basic definitions like orders on sets and multisets (Section 3.2), orders on terms (Section 3.3) and finally notion of ordering constraints (Section 3.4) where we also overview some known results on solving ordering constraints.

## 3.1 Term algebras

The main objects we will be working with are terms over a finite signature. A *signature* is a finite set of function symbols with assigned arities (nonnegative integers) e.g. $\Sigma = \{g(,), h(), c\}$ is a signature with function symbols $g$ of arity two, (such symbols also called binary symbols), $h$ of arity one,(also called unary symbols) and $c$ of arity zero (also called constants). We will denote a signature by $\Sigma$. *Terms* of the signature $\Sigma$ over a set of variables $X$ are defined by induction as follows, constants and variables are terms, and for each function symbol $g \in \Sigma$ of a positive arity $n$ and terms $t_1, \dots t_n$ we have $g(t_1, \dots, t_n)$ is a term. Terms which contain no variables are called *ground terms*.

Definition 3.1.1 (substitution) A *substitution* $\theta$ is a mapping from the set of variables $X$ to the set of terms. This mapping can be extended from variables to terms in the following canonical way. For every constant $c$, $\theta(c) = c$ and for every nonconstant term $g(t_1, \dots, t_n)$, $\theta(g(t_1, \dots, t_n)) = g(\theta(t_1), \dots, \theta(t_n))$. In the sequel we consider substitutions which are identity on all but finitely many variables. An application of a substitution $\theta$ to a term $t$ will be denoted by $t\theta$. $\square$

Definition 3.1.2 Let $\Sigma$ be a finite signature which contains at least one constant. The $\Sigma$–*term algebra* $\mathrm{TA}(\Sigma)$ is an algebra with the domain of the set of all ground $\Sigma$–terms and the following interpretation: constants interpreted by constants from the domain and the value of a function $g$ on terms $\bar{t}$ is the term $g(\bar{t})$. $\square$

When the signature is clear from the context, then we say the term algebra instead of the $\Sigma$–term algebra.

Some authors call the $\Sigma$–term algebra as an absolutely free algebra (in the class of all $\Sigma$–algebras), it means that there exists a unique homomorphism from the term algebra into any $\Sigma$–algebra.

In the future we always assume that our signature contains at least one constant symbol.

## 3.2 Orders on sets

Definition 3.2.1 A *partially ordered set* $(A, \geq)$ is a set $A$ with a binary relation $\geq$ which is reflexive, transitive and antisymmetric. An order is called *linear* or *total* if for any two elements $a, b \in A$ either $a \geq b$ or $b \geq a$. We say that $a$ is *strictly greater* than $b$, denoting $a > b$, if $a \geq b$ and $b \not\geq a$. An order is called *well-founded* if there is no infinite decreasing chain $a > b > \cdots$. $\square$

Let us define multisets which is a generalization of sets (for the properties of multisets see [Baader and Nipkow 1998]).

Definition 3.2.2 A *multiset* $M$ over a set $A$ is a function $M : A \to \mathbb{N}$. $\square$

A multiset is finite if there are only finitely many $x$ such that $M(x) > 0$. We will consider only finite multisets. We adopt a standard set notation for multisets for example $\dot{\{}a, a, b\dot{\}}$ denotes the multiset $M = \{a \to 2, b \to 1\}$ and we write $a \mathrel{\dot{\in}} M$ if $M(a) > 0$. The union, intersection and multiset difference are defined as follows: $(M_1 \cup M_2)(x) = M_1(x) + M_2(x)$, $(M_1 \cap M_2)(x) = min(M_1(x), M_2(x))$ and $(M_1 \dot{-} M_2)(x) = max(0, M_1(x) - M_2(x))$.

One of the important properties of multisets is as follows: if have a total, well–founded order on a set $A$ then we can extend this order into a total, well–founded order on the multisets over $A$.

DEFINITION 3.2.3 Given a strict order $>$ on a set $A$, we define the corresponding *multiset order* on all multisets over $A$ as follows: $M >_{mul} N$ if there exist multisets $X, Y$ such that the following holds:

- $\emptyset \neq X \subseteq M$, and
- $N = (M \setminus X) \cup Y$, and
- $\forall y \in Y \exists x \in X \quad x > y$.

□

We also call $>_{mul}$ as a *multiset extension* of $>$. The multiset orders were introduced by Dershowitz and Manna [1979].

PROPOSITION 3.2.4 *If $>$ is a strict order then $>_{mul}$ is a strict order. If $>$ is a well–founded order then $>_{mul}$ is a well–founded order.* □

For a proof let us refer to [Baader and Nipkow 1998].

## 3.3 Orders on terms

One of the most general classes of orders on terms which is used in automated deduction is so-called simplification orders introduced by Dershowitz [1979].

DEFINITION 3.3.1 A strict order $>$ on $\mathrm{TA}(\Sigma)$ is called a *simplification order* if it has the following properties:

- $>$ is *monotone* (or *compatible with $\Sigma$–operations*): for all $s_1, s_2 \in \mathrm{TA}(\Sigma)$ and $n$–ary function symbol $g \in \Sigma$, $s_1 > s_2$ implies $g(t_1, \ldots, t_{i-1}, s_1, t_{i+1}, \ldots, t_n) > g(t_1, \ldots, t_{i-1}, s_2, t_{i+1}, \ldots, t_n)$ for all $i$, $1 \leq i \leq n$, and all $t_1, \ldots, t_{i-1}, t_{i+1}, \ldots, t_n \in \mathrm{TA}(\Sigma)$.
- $>$ has a *subterm property*: if $r[s] \neq s$, then $r[s] > s$.

□

One of the main properties of simplification orders is that every simplification order is well-founded [Dershowitz 1979].

There are two subclasses of simplification orders that are widely used because of a possibility to generate them automatically for a given set of clauses. They are: Knuth-Bendix orders and recursive path orders.

Let us start from the definition of recursive path orders. Recursive path orders are generalization of lexicographic path orders introduced by Kamin and Lévy [1980] and multiset path orders introduced by Dershowitz [1982].

DEFINITION 3.3.2 Let us fix a strict order $\gg$ on $\Sigma$. The *lexicographic path order* $>_{lpo}$ on TA($\Sigma$) induced by $\gg$ is defined as follows: $g(s_1, \ldots, s_n) >_{lpo} h(t_1, \ldots, t_m)$ if one of the following conditions holds:

- $s_i \geq_{lpo} h(t_1, \ldots, t_m)$ for some $i$, $1 \leq i \leq n$.
- $g \gg h$ and $g(s_1, \ldots, s_n) >_{lpo} t_i$ for all $i = 1, \ldots, m$.
- $g = h$ and $g(s_1, \ldots, s_n) >_{lpo} t_i$ for all $i = 1, \ldots, m$ and there exists $j$, $1 \leq j \leq m$, such that $s_1 = t_1, \ldots, s_{j-1} = t_{j-1}$ and $s_i >_{lpo} t_i$.

□

Lexicographic path orders are simplification orders (for a proof see [Baader and Nipkow 1998]).

REMARK 3.3.3 If our signature contains at least two non-constant function symbols then there are terms with an infinite number of different terms below them with respect to the lexicographical path order.

PROOF. We illustrate the proof for the case $\Sigma = \{g(), h(), c\}$ and $g \gg h$ the general case is similar. It is easy to check that all terms $h^n(c)$ are strictly less than the term $g(c)$ for any natural number $n$. □

One of the main usage of orders in automated deduction is to replace "big" terms by "smaller" terms. This remark shows that "small" terms in the sense of lexicographic path orders can be arbitrarily large in the physical representation. We will see later, Lemma 3.3.9, that for a rather large class of Knuth-Bendix orders the number of terms below any fixed term is finite.

Let us consider multiset path orders introduced by Dershowitz [1982]. These orders are defined on the equivalence classes over the multiset equivalence. The *multiset equivalence* $=_{mul}$ is the least equivalence relation such that if we have that a term $t$ is in the equivalence class, then any term obtained by permutation of immediate subterms of $t$ is in the same equivalence class.

Definition 3.3.4 Let us fix a strict order $\gg$ on $\Sigma$. *The multiset path order* $>_{mpo}$ on $\mathrm{TA}(\Sigma)$ induced by $\gg$ is defined as follows $g(s_1, \ldots, s_n) >_{mpo} h(t_1, \ldots, t_m)$ if one of the following conditions holds:

- $s_i >_{mpo} h(t_1, \ldots, t_m)$ or $s_i =_{mul} h(t_1, \ldots, t_m)$ for some $i$, $1 \leq i \leq n$.
- $g \gg h$ and $g(s_1, \ldots, s_n) >_{mpo} t_i$ for all $i = 1, \ldots, m$.
- $g = h$ and $\dot{\{}s_1, \ldots, s_n\dot{\}} >_{mul} \dot{\{}t_1, \ldots, t_n\dot{\}}$.

□

Recursive path orders on terms is a combination of lexicographical path orders and multiset path orders. We divide our signature $\Sigma$ into two disjoint sets $\Sigma_{lex}$ and $\Sigma_{mul}$. The multiset equivalence $=_{mul}$ on $\mathrm{TA}(\Sigma_{lex} \cup \Sigma_{mul})$ is defined w.r.t. function symbols in $\Sigma_{mul}$. That is, $=_{mul}$ is the least equivalence relation such that if we have that a term $t$, with top function symbol in $\Sigma_{mul}$, is in the equivalence class, then any term obtained by permutation of immediate subterms of $t$ is in the same equivalence class.

Definition 3.3.5 Let us fix a strict order $\gg$ on $\Sigma$.

The *recursive path order* $>_{rpo}$ on $\mathrm{TA}(\Sigma)$ induced by $\gg$ is defined as follows: $g(s_1, \ldots, s_n) >_{rpo} h(t_1, \ldots, t_m)$ if one of the following conditions holds:

- $s_i >_{rpo} h(t_1, \ldots, t_m)$ or $s_i =_{mul} h(t_1, \ldots, t_m)$ for some $i$, $1 \leq i \leq n$.
- $g \gg h$ and $h(s_1, \ldots, s_n) >_{rpo} t_i$ for all $i = 1, \ldots, m$.
- $g = h$, $g \in \Sigma_{lex}$ and $g(s_1, \ldots, s_n) >_{rpo} t_i$ for all $i = 1, \ldots, m$ and there exists $j$, $1 \leq j \leq m$, such that $s_1 = t_1, \ldots, s_{j-1} = t_{j-1}$ and $s_i >_{rpo} t_i$.
- $g = h$, $g \in \Sigma_{mul}$ and $\dot{\{}s_1, \ldots, s_n\dot{\}} >_{mul} \dot{\{}t_1, \ldots, t_n\dot{\}}$.

□

Lexicographic and multiset path orders are the special cases of recursive path orders, when we fix $\Sigma_{lex} = \Sigma$, $\Sigma_{mul} = \Sigma$ respectively. Recursive path orders on terms, modulo the multiset equivalence, are well–founded, compatible with $\Sigma$–operations, and total.

Let us now define Knuth-Bendix orders on $\mathrm{TA}(\Sigma)$ [Knuth and Bendix 1970]. Knuth-Bendix orders is a family of orders parameterized by two parameters: a weight function and a precedence relation.

Definition 3.3.6 (weight function) We call a *weight function* on $\Sigma$ any function $w : \Sigma \to \mathbb{N}$ such that (i) $w(a) > 0$ for every constant $a \in \Sigma$, (ii) there exist at most one unary function symbol $f \in \Sigma$ such that $w(f) = 0$. Given a weight function $w$, we call $w(g)$ the *weight* of $g$. The *weight* of any ground term $t$, denoted $|t|$, is defined as follows: for every constant $c$ we have $|c| = w(c)$ and for every function symbol $g$ of a positive arity we have $|g(t_1, \ldots, t_n)| = w(g)+|t_1|+\ldots+|t_n|$. □

Definition 3.3.7 (precedence relation) A *precedence relation* on $\Sigma$ is any total order $\gg$ on $\Sigma$. A precedence relation $\gg$ is said to be *compatible* with a weight function $w$ if, whenever $f$ is a unary function symbol $f$ of weight zero, $f$ is the greatest element w.r.t. $\gg$. □

These conditions on the weight function and precedence relation ensure that every Knuth-Bendix order is a simplification order total on ground terms (see, e.g. [Baader and Nipkow 1998]).

Let us consider a weight function $w$ on $\Sigma$ and a precedence relation $\gg$ on $\Sigma$, compatible with $w$.

Definition 3.3.8 The *Knuth-Bendix order* on TA($\Sigma$) is the binary relation $\succ_{KBO}$ defined as follows. For any ground terms $t = g(t_1, \ldots, t_n)$ and $s = h(s_1, \ldots, s_k)$ we have $t \succ_{KBO} s$ if one of the following conditions holds:

1. $|t| > |s|$;
2. $|t| = |s|$ and $g \gg h$;
3. $|t| = |s|$, $g = h$ and for some $1 \leq i \leq n$ we have $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i \succ_{KBO} s_i$.

□

Note that every Knuth-Bendix order is a total monotonic well-founded order, see, e.g. [Baader and Nipkow 1998].

For a unary function symbol $f$ and a term $t$, let $f^m(t)$ denote a term obtained by $m$ applications of $f$ to $t$. Let us prove the following simple properties of weight functions which we will use later.

Lemma 3.3.9 *Every weight function satisfies the following properties.*

1. *The weight of every term is positive.*

2. *If $\Sigma$ contains the unary function symbol of the weight* $0$*, then, for every weight, either there are no terms of that weight or there are infinitely many.*

3. *If a term $s$ is a subterm of $t$ then either $|t| > |s|$ or $|t| = |s|$ and $t$ has the form $f^m(s)$ for some $m \geq 0$, where $f$ is the unary function symbol of the weight* $0$.

4. *If $\Sigma$ contains no unary function symbol of the weight* $0$*, then for every natural number $n$ there is only a finite number of terms of the weight $n$.*

Proof. First property follows from the fact that the weight of every constant is positive.

Denote the unary function symbol of the weight $0$ as $f$. Then the second property follows from the fact that if we have a term $t$ then for every $m \in \mathbb{N}$ the term $f^m(t)$ has the same weight as $t$.

To prove the last two properties let us show that if we consider a nonconstant term $t$ with a top function symbol different from $f$ then the weight of $t$ is strictly greater than the weight of any of its immediate subterms.

Indeed, if we consider a term $t = g(t_1, \ldots, t_n)$ where $g$ is different from $f$, then $|g(t_1, \ldots, t_n)| = w(g) + |t_1| + \cdots + |t_n|$ and all possible cases are as follows:

- either $w(g) > 0$ and $|t| > |t_1| + \cdots + |t_n|$ and therefore $|t| > |t_i|$ for $1 \leq i \leq n$, or

- $w(g) = 0$ and $n > 1$ so we have $|t| = |t_1| + \ldots + |t_n|$, and since the weight of every term is positive we have that $|t| > |t_i|$ for $1 \leq i \leq n$.

From this, the third property follows immediately. To show the last property consider a signature without the unary function symbol of zero weight. From the observation above we have that in this case each term has depth less or equal than its weight. Since there are only finite number of terms of a fixed depth we conclude that for each weight there is only a finite number of terms of this weight. $\square$

From this lemma it follows that if our signature contains no unary function symbol of weight zero then there is only a finite number of terms below each term. The following example shows that if our signature contains the unary function symbol of the weight 0 and a binary functional symbol then there exists a term $t$ with an infinite number of terms below it of the same weight as $t$.

EXAMPLE 3.3.10 Let $\Sigma = \{f(), g(,), c_1\}$ and $f \gg g \gg c_1$ and $w(f) = 0$. Then $g(f(c_1), c_1) \succ_{KBO} g(c_1, f^n(c_1))$ for any natural number $n$. Moreover we have that $w(g(f(c_1), c_1)) = w(g(c_1, f^n(c_1)))$ for any natural number $n$. $\square$

## 3.4 Ordering constraints

In this section we describe types of ordering constraints that we will work with.

Let us fix a signature $\Sigma$ which induces the term algebra $\mathrm{TA}(\Sigma)$ and let us fix an order on this term algebra. We denote $\mathrm{TA}_>(\Sigma)$ the structure of the term algebra with the order $>$ and we call this structure an *ordered term algebra*.

DEFINITION 3.4.1 A *conjunctive ordering constraint* (or just *a constraint* ) is a conjunction of atomic formulas of the language of $\mathrm{TA}_>(\Sigma)$. $\square$

For example, if we have $\Sigma = \{h(,), g(), c\}$ then $h(x, g(y)) > c \wedge g(x) > h(g(z), y) \wedge g(g(y)) = g(g(c))$ is a constraint with free variables $x, y$.

DEFINITION 3.4.2 A *quantifier–free constraint* is a quantifier–free formula of the language of $\mathrm{TA}_>(\Sigma)$. $\square$

For example, if we have $\Sigma = \{h(,), g(), c\}$ then $(h(g(y), z) > z) \rightarrow \neg(z = g(m) \vee z > g(c))$ is a quantifier–free constraint.

DEFINITION 3.4.3 A *first–order constraint* is a first order formula of the language of $\mathrm{TA}_>(\Sigma)$. $\square$

For example, if we have $\Sigma = \{h(,), g(), c\}$ then $\forall y \exists z(h(g(y), x) > z \wedge y > c)$ is a first–order constraint.

A constraint $\phi(\bar{x})$ is *satisfiable* in the ordered term algebra $\mathrm{TA}_>(\Sigma)$ if $\mathrm{TA}_>(\Sigma) \models \exists \bar{x} \phi(\bar{x})$ i.e. there exist ground terms $\bar{t}$ such that the sentence $\phi(\bar{t})$ is valid in our ordered term algebra. Let us fix an ordered term algebra $\mathrm{TA}_>(\Sigma)$ then the *constraint satisfiability problem* is a problem to decide for a given constraint whether it satisfiable in $\mathrm{TA}_>(\Sigma)$ or not. A *solution* to a constraint is a substitution which makes this constraint valid. It is clear, that the quantifier–free (first–order) constraint satisfiability problem is equivalent to the problem of the decidability of the existential (first–order) theory of the ordered term algebra.

## 3.5 Solving ordering constraints

In this section we overview some results on solving recursive path ordering constraints.

Term algebras are rather well-studied structures. Maĺcev [1961] was the first to prove the decidability of the first-order theory of term algebras. Other methods of proving decidability were developed by Comon and Lescanne [1989], Kunen [1987], Belegradek [1988] and Maher [1988]. The complexity of the first-order theory of any term algebra over a signature containing a binary function symbol is nonelementary, i.e. not bounded by any tower of exponents $2^{\cdot^{\cdot^{2^n}}}$ (see [Ferrante and Rackoff 1979]).

If we introduce a binary predicate into a term algebra, then one can obtain a richer theory. Term algebras with the subterm predicate have an undecidable first–order theory and a decidable existential theory [Venkataraman 1987].

Let us consider term algebras with lexicographic path orders.

THEOREM 3.5.1 *[Comon 1990] The quantifier–free constraint satisfiability problem for lexicographic path orders is decidable.* □

Later, it was shown that this problem is NP–complete.

THEOREM 3.5.2 *[Nieuwenhuis 1993] The quantifier–free constraint satisfiability problem for lexicographic path orders is NP–complete.* □

Let us prove a simple result (similar to the result from [Nieuwenhuis 1993]) from which NP-hardness will follow.

PROPOSITION 3.5.3 *For any structure $S$ with at least two elements the following holds.*

1. *The problem of deciding whether a given existential formula is valid in $S$ is NP-hard.*

2. *The problem of deciding whether a given first-order formula is valid in $S$ is PSPACE-hard.*

PROOF. It is well-known that the problem of satisfiability of propositional formulas is NP-complete and the problem of satisfiability of quantified propositional formulas is PSPACE-complete (see e.g. [Papadimitriou 1994]). We show how to reduce satisfiability of propositional formulas to satisfiability of quantifier-free constraints in $S$ and satisfiability of quantified propositional formulas to to satisfiability of first-order constraints in $S$.

We transform any propositional formula $P$ into a quantifier-free constraint $C$ as follows. For any propositional variable $X$ occurring into $P$ we fix a pair of new variables $x_1, x_2$. Any occurrence of a propositional variable $X$ we replace with the formula $x_1 = x_2$. It is easy to check that the obtained constraint $C$ is satisfiable in $S$ if and only if $P$ is satisfiable.

For quantified propositional formulas, in addition to the previous transformations, we replace each propositional quantifier $\exists X$ with first-order quantifiers $\exists x_1 \exists x_2$, likewise $\forall X$ we replace with $\forall x_1 \forall x_2$. It is easy to check that the obtained first-order constraint is satisfiable in $S$ if and only if the initial propositional quantified formula is satisfiable. $\square$

COROLLARY 3.5.4 *If a term algebra contains at least two elements then the quantifier-free constraint satisfiability problem is NP-hard and first-order constraint satisfiability problem is PSPACE-hard, for any order.* $\square$

It turns out that for lexicographic path orders even the problem of satisfiability of the atomic formulas is NP-complete [Comon and Treinen 1994].

Although the constraint satisfiability problem for lexicographic path orders is in NP, a practical algorithm was presented only in [Nieuwenhuis and Rivero 1999].

Let us consider first–order lexicographic path ordering constraints. Treinen [1990] proved the undecidability of the constraint satisfiability problem for a generalization of lexicographical path orders. He used a reduction of the Post correspondence problem to the first–order constraint satisfiability problem. Later, Comon and Treinen [1997] proved that the constraint satisfiability problem for lexicographic path orders is undecidable again using a reduction of the Post correspondence problem.

THEOREM 3.5.5 *[Comon and Treinen 1997] Let us fix a signature $\Sigma$ and an order $\gg$ on $\Sigma$ such that there exists a binary function $h$ minimal with respect to $\gg$. This order induces a lexicographic order $\succ$ on $\mathrm{TA}(\Sigma)$ such that the first-order theory of the ordered term algebra $\mathrm{TA}_{\succ}(\Sigma)$ is undecidable.* $\square$

It turns out that if we consider a signature which consists only of constants and unary function symbols then the problem of satisfiability of first–order lexicographic path ordering constraints is decidable [Narendran and Rusinowitch 2000].

Let us consider recursive path ordering constraints. The quantifier-free constraints satisfiability problem is shown to be decidable [Jouannaud and Okada 1991] and NP–complete [Narendran et al. 1998]. To our knowledge it is unknown whether the satisfiability problem of first-order multiset path ordering constraints is decidable or not.

# Chapter 4

# Knuth-Bendix constraint solving is NP-complete

This chapter is based on papers [Korovin and Voronkov 2000, Korovin and Voronkov 2001*a*].

In this chapter we present a nondeterministic polynomial-time algorithm for solving Knuth-Bendix ordering constraints, and hence show that the problem is contained in NP for every term algebra with a Knuth-Bendix order. As a consequence, we obtain that the existential first-order theory of any term algebra with a Knuth-Bendix order is NP-complete too. Let us note that the problem of solvability of a Knuth-Bendix ordering constraints consisting of a single inequality can be solved in polynomial time see Chapter 6.

This chapter is structured as follows. In Section 4.2 we introduce the notion of isolated form of constraints and show that every constraint can be effectively transformed into an equivalent disjunction of constraints in isolated form. This transformation is represented as a nondeterministic polynomial-time algorithm computing members of this disjunction. After this, it remains to show that solvability of constraints in isolated form can be decided by a nondeterministic polynomial-time algorithm. In Section 4.3 we present such an algorithm using transformation to systems of linear Diophantine inequalities over the weights of variables. Finally, in Section 4.4 we complete the proof of the main result and present some examples.

## 4.1 Preliminaries

In this chapter, $f$ *will always denote a unary function symbol of the weight* $0$.

In the sequel we assume a fixed weight function $w$ on $\Sigma$ and a fixed precedence relation $\gg$ on $\Sigma$, compatible with $w$.

The main result of this chapter is the following.

> **Theorem 4.4.2:** *The existential first-order theory of any term algebra with the Knuth-Bendix order in a signature with at least two symbols is NP-complete.*

To prove this result, we introduce a notion of Knuth-Bendix ordering constraint and show the following.

> **Theorem 4.4.1:** *For every Knuth-Bendix order, the problem of solving ordering constraints is contained in NP.*

We also show that the systems of linear Diophantine equations and inequalities can be represented as ordering constraints for some Knuth–Bendix orders, and as a corollary we obtain the following.

> **Theorem 4.4.4:** *For some Knuth-Bendix orders, the problem of solving ordering constraints is NP-complete.*

Some authors [Martin 1987, Baader and Nipkow 1998] define Knuth-Bendix orders with real-valued weight functions. We do not consider such orders here, because for real-valued functions even the comparison of ground terms can be undecidable (see Example 4.4.7 in Section 4.4). Sometimes it is useful to consider constraint solving problem for the so-called extended signature semantics, where we look for solutions to the constraints in some possible extension of the signature. For recursive path orders this problem is studied in [Nieuwenhuis 1993, Nieuwenhuis and Rivero 1999]. A possible direction for future research is to apply the methods of this chapter for solving Knuth-Bendix ordering constraints in the extended signature semantics.

The proof of Theorem 4.4.2 will be given after a series of lemmas. The idea of the proof is as follows. First, we will make $\mathrm{TA}(\Sigma)$ into a two-sorted structure by adding the sort of natural numbers, and extend its signature by

1. the weight function $|\cdot|$ on ground terms;

2. the addition function $+$ on natural numbers;

3. the Knuth-Bendix order $\succ_{KBO}$ on ground terms.

Given an existential formula of the first-order theory of a term algebra with the Knuth-Bendix order, we will transform it step by step into an equivalent disjunction of existential formulas of the extended signature. The main aim of these steps is to replace all occurrences of $\succ_{KBO}$ by linear Diophantine inequalities on the weights of variables. After such a transformation we will obtain existential formulas consisting of linear Diophantine inequalities on the weight of variables plus statements expressing that, for some fixed natural number $N$, there exists at least $N$ terms of the same weight as $|x|$, where $x$ is a variable. We will show how these statements can be expressed using systems of linear Diophantine inequalities on the weights of variables and then use the fact that the decidability of systems of linear Diophantine equations is in NP.

We denote by $\mathrm{TA}^+(\Sigma)$ the following structure with two sorts: the *term algebra sort* and the *arithmetical sort*. The domains of the term algebra sort and the arithmetical sort are the sets of ground terms of $\Sigma$ and natural numbers, respectively. The signature of $\mathrm{TA}^+(\Sigma)$ consists of

1. all symbols of $\Sigma$ interpreted as in $\mathrm{TA}(\Sigma)$;

2. symbols $0, 1, >, +$ having their conventional interpretation over natural numbers;

3. the binary relation symbol $\succ_{KBO}$ on the term algebra sort, interpreted as the Knuth-Bendix order;

4. the unary function symbol $|\cdot|$, interpreted as the weight function mapping terms to numbers.

When we need to distinguish the equality $=$ on the term algebra sort from the equality on the arithmetical sort, we denote the former by $=_{\mathrm{TA}}$, and the latter by $=_{\mathbb{N}}$.

We will prove that the existential theory of $\mathrm{TA}^+(\Sigma)$ is in NP, from which the fact that the existential theory of any term algebra with the Knuth-Bendix order belongs to NP follows immediately. We consider *satisfiability*, *validity*, and *equivalence* of formulas with respect to the structure $\mathrm{TA}^+(\Sigma)$. We call a *constraint* in the language of $\mathrm{TA}^+(\Sigma)$ any conjunction of atomic formulas of this language.

LEMMA 4.1.1 *The existential theory of* $\mathrm{TA}^+(\Sigma)$ *is in NP if and only if so is the constraint satisfiability problem.*

PROOF. Obviously any instance $A$ of the constraint satisfiability problem can be considered as validity of the existential sentence $\exists x_1 \ldots x_n A$, where $x_1, \ldots, x_n$ are all variables of $A$, so the "only if" direction is trivial.

To prove the "if" direction, take any existential formula $\exists x_1, \ldots, x_n A$. This formula is satisfiable if and only if so is the quantifier-free formula $A$. By converting $A$ into disjunctive normal form we can assume that $A$ is built from literals using $\wedge, \vee$. Replace in $A$

1. any formula $\neg s \succ_{KBO} t$ by $s =_{\mathrm{TA}} t \vee t \succ_{KBO} s$,
2. any formula $\neg s =_{\mathrm{TA}} t$ by $s \succ_{KBO} t \vee t \succ_{KBO} s$,
3. any formula $\neg p > q$ by $p =_{\mathbb{N}} q \vee q > p$,
4. any formula $\neg p =_{\mathbb{N}} q$ by $p > q \vee q > p$,

and convert $A$ into disjunctive normal form again. It is easy to see that we obtain a disjunction of constraints. The transformation gives an equivalent formula since both orders $\succ_{KBO}$ and $>$ are total.

It follows from these arguments that there exists a nondeterministic polynomial-time algorithm which, given an existential sentence $A$, computes on every branch a constraint $C_i$ such that $A$ is valid if and only if one of the constraints $C_i$ is satisfiable. $\square$

A substitution $\theta$ is called *grounding* for an expression $C$ (i.e., term or constraint) if for every variable $x$ occurring in $C$ the term $\theta(x)$ is ground. Let $\theta$ be a substitution grounding for an expression $C$. We denote by $C\theta$ the expression obtained from $C$ by replacing in it every variable $x$ by $\theta(x)$. A substitution $\theta$ is called a *solution* to a constraint $C$ if $\theta$ is grounding for $C$ and $C\theta$ is valid in $\mathrm{TA}^+(\Sigma)$.

In the sequel we will often replace a constraint $C(\bar{x})$ by a formula $A(\bar{x}, \bar{y})$ containing extra variables $\bar{y}$ and say that they are "equivalent". By this we mean that $\mathrm{TA}^+(\Sigma) \models \forall \bar{x}(C(\bar{x}) \leftrightarrow \exists \bar{y} A(\bar{x}, \bar{y}))$. In other words, the set of solutions to $C$ is exactly the set solutions to $A$ projected on $\bar{x}$.

## 4.2 Isolated forms

We are interested not only in satisfiability of constraints, but also in their solutions. Our algorithm will consist of equivalence-preserving transformation steps. When the signature contains no unary function symbol of the weight 0, the transformation will preserve equivalence in the following strong sense. At each step, given a constraint $C(\bar{x})$, we transform it into constraints $C_1(\bar{x}, \bar{y}), \ldots, C_n(\bar{x}, \bar{y})$ such that for every sequence of ground terms $\bar{t}$, the constraint $C(\bar{t})$ holds if and only if there exist $k$ and a sequence of ground terms $\bar{s}$ such that $C_k(\bar{t}, \bar{s})$ holds. In other words, the following formula holds in $\mathrm{TA}^+(\Sigma)$:

$$C(\bar{x}) \leftrightarrow \exists \bar{y}(C_1(\bar{x}, \bar{y}) \vee \ldots \vee C_n(\bar{x}, \bar{y})).$$

Moreover this transformations will be presented as a nondeterministic polynomial-time algorithm which computes on every branch some $C_i(\bar{x}, \bar{y})$, and every $C_i(\bar{x}, \bar{y})$ is computed on at least one branch. When the signature contains a unary function symbol of the weight 0, the transformation will preserve a weaker form of equivalence: some solutions will be lost, but solvability will be preserved. More precisely, we will introduce a notion of an $f$-variant of a term and show that the following formula holds:

$$C(\bar{x}) \leftrightarrow \exists \bar{y} \exists \bar{z}(\textit{f-variant}(\bar{x}, \bar{z}) \wedge (C_1(\bar{z}, \bar{y}) \vee \ldots \vee C_n(\bar{z}, \bar{y}))), \tag{4.1}$$

where $\textit{f-variant}(\bar{x}, \bar{z})$ expresses that $\bar{x}$ and $\bar{z}$ are $f$-variants.

In our proof, we will reduce solvability of Knuth-Bendix ordering constraints to the problem of solvability of systems of linear Diophantine inequalities on the weights of variables. Condition 1 in Definition 3.3.8 of the Knuth-Bendix order, $|t| > |s|$ has a simple translation into a linear Diophantine inequality, but conditions 2 and 3 do not have. So we will split the Knuth-Bendix order in two partial orders: $\succ_w$ corresponding to condition 1 and $\succ_{lex}$ corresponding to conditions 2 and 3. Formally, we denote by $t \succ_w s$ the formula $|t| > |s|$ and by $t \succ_{lex} s$ the formula $|t| =_{\mathbb{N}} |s| \wedge t \succ_{KBO} s$. Obviously, $t_1 \succ_{KBO} t_2$ if and only if $t_1 \succ_{lex} t_2 \vee t_1 \succ_w t_2$. So in the sequel we will assume that $\succ_{KBO}$ is replaced by the new symbols $\succ_{lex}$ and $\succ_w$.

We use $x_1 \succ_{KBO} x_2 \succ_{KBO} \ldots \succ_{KBO} x_n$ to denote the formula $x_1 \succ_{KBO} x_2 \wedge x_2 \succ_{KBO} x_3 \wedge \ldots \wedge x_{n-1} \succ_{KBO} x_n$, and similar for other binary symbols in place of $\succ_{KBO}$.

A term $t$ is called *flat* if $t$ is either a variable or has the form $g(x_1, \ldots, x_m)$, where $g \in \Sigma$, $m \geq 0$, and $x_1, \ldots, x_m$ are variables. We call a constraint *chained* if

1. it has a form $t_1 \# t_2 \# \ldots \# t_n$, where each occurrence of $\#$ is $\succ_w$, $\succ_{lex}$ or $=_{\mathrm{TA}}$;

2. each term $t_i$ is flat;

3. if some of the $t_i$'s has the form $g(x_1, \ldots, x_n)$, then $x_1, \ldots, x_n$ are some of the $t_j$'s.

For example $g(x, y) \succ_w f(y) \succ_{lex} y \succ_w x =_{\mathrm{TA}} z$ is a chained constraint.

Denote by $\perp$ the logical constant "false".

LEMMA 4.2.1 *Any constraint $C$ is equivalent to a disjunction $C_1 \vee \ldots \vee C_k$ of chained constraints. Moreover, there exists a nondeterministic polynomial-time algorithm which, for a given $C$, computes on every branch either $\perp$ or some $C_i$; and every $C_i$ is computed on at least one branch.*

PROOF. First, we can apply flattening to all terms occurring in $C$ as follows. If a nonflat term $g(t_1, \ldots, t_m)$ occurs in $C$, take any $i$ such that $t_i$ is not a variable. Then replace $C$ by $v = t_i \wedge C'$, where $v$ is a new variable and $C'$ is obtained from $C$ by replacing all occurrences of $t_i$ by $v$. After a finite number of such replacements all terms will become flat.

Let $s, t$ be flat terms occurring in $C$ such that no comparison $s \# t$ occurs in $C$. Using the valid formula $s \succ_w t \vee s \succ_{lex} t \vee s =_{\mathrm{TA}} t \vee t \succ_w s \vee t \succ_{lex} s$ we can replace $C$ by the disjunction of the constraints

$$
\begin{array}{lll}
s \succ_w t \wedge C, & s \succ_{lex} t \wedge C, & s =_{\mathrm{TA}} t \wedge C, \\
t \succ_w s \wedge C, & t \succ_{lex} s \wedge C. &
\end{array}
$$

By repeatedly doing this transformation we obtain a disjunction of constraints $C_1 \vee \ldots \vee C_k$ in which for every $i \in \{1, \ldots, k\}$ and every terms $s, t$ occurring in $C_i$, some comparison constraint $s \# t$ occurs in $C_i$.

To complete the proof we show how to turn each $C_i$ into a chained constraint. Let us call a *cycle* any constraint $s_1 \# s_2 \# \ldots \# s_n \# s_1$, where $n \geq 1$. We can remove all cycles from $C_i$ using the following observation:

1. if all $\#$ in the cycle are $=_{\mathrm{TA}}$, then $s_n \# s_1$ can be removed from the constraint;

2. if some $\#$ in the cycle is $\succ_w$ or $\succ_{lex}$, then the constraint $C_i$ is unsatisfiable.

After removal of all cycles the constraint $C_i$ can still be not chained because it can contain *transitive subconstraints* of the form $s_1 \# s_2 \# \ldots \# s_n \wedge s_1 \# s_n$, $n \geq 2$. Then either $C_i$ is unsatisfiable or $s_1 \# s_n$ can be removed using the following observations:

1. *Case: $s_1 \# s_n$ is $s_1 \succ_w s_n$.* If some $\#$ in $s_1 \# s_2 \# \ldots \# s_n$ is $\succ_w$, then $s_1 \succ_w s_n$ follows from $s_1 \# s_2 \# \ldots \# s_n$, otherwise $s_1 \# s_2 \# \ldots \# s_n$ implies $|s_1| = |s_n|$ and hence $C_i$ is unsatisfiable.

2. *Case: $s_1 \# s_n$ is $s_1 \succ_{lex} s_n$.* If some $\#$ in $s_1 \# s_2 \# \ldots \# s_n$ is $\succ_w$, then $C_i$ is unsatisfiable. If all $\#$ in $s_1 \# s_2 \# \ldots \# s_n$ are $=_{\mathrm{TA}}$, then $C_i$ is unsatisfiable too. Otherwise, all $\#$ in $s_1 \# s_2 \# \ldots \# s_n$ are either $\succ_{lex}$ or $=_{\mathrm{TA}}$, and at least one of them is $\succ_{lex}$. It is not hard to argue that $s_1 \succ_{lex} s_n$ follows from $s_1 \# s_2 \# \ldots \# s_n$.

3. *Case: $s_1 \# s_n$ is $s_1 =_{\mathrm{TA}} s_n$.* If all $\#$ in $s_1 \# s_2 \# \ldots \# s_n$ are $=_{\mathrm{TA}}$, then $s_1 =_{\mathrm{TA}} s_n$ follows from $s_1 \# s_2 \# \ldots \# s_n$, otherwise $C_i$ is unsatisfiable.

It is easy to see that after the removal of all cycles and transitive subconstraints the constraint $C_i$ becomes chained.

Note that the transformation of $C$ into the disjunction of constraints $C_1 \vee \ldots \vee C_k$ in the proof can be done in nondeterministic polynomial time in the following sense: there exists a nondeterministic polynomial-time algorithm which, given $C$, computes on every branch either $\perp$ or some $C_i$, and every $C_i$ is computed on at least one branch. $\square$

We will now introduce several special kinds of constraints which will be used in our proofs below, namely *arithmetical*, *triangle*, *simple*, and *isolated*.

A constraint is called *arithmetical* if it uses only arithmetical relations $=_{\mathbb{N}}$ and $>$, for example $|f(x)| > |a| + 3$.

A constraint $y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_n =_{\mathrm{TA}} t_n$ is said to be in *triangle form* if

1. $y_1, \ldots, y_n$ are pairwise different variables, and

2. for all $j \geq i$ the variable $y_i$ does not occur in $t_j$.

The variables $y_1, \ldots, y_n$ are said to be *dependent* in this constraint.

A constraint is said to be *simple* if it has the form

$$x_{11} \succ_{lex} x_{12} \succ_{lex} \ldots \succ_{lex} x_{1n_1} \wedge \ldots \wedge x_{k1} \succ_{lex} x_{k2} \succ_{lex} \ldots \succ_{lex} x_{kn_k},$$

where $x_{11}, \ldots, x_{kn_k}$ are pairwise different variables.

A constraint is said to be in *isolated form* if either it is $\bot$ or it has the form

$$C_{arith} \wedge C_{triang} \wedge C_{simp},$$

where $C_{arith}$ is an arithmetical constraint, $C_{triang}$ is in triangle form, and $C_{simp}$ is a simple constraint such that no variable of $C_{simp}$ is dependent in $C_{triang}$.

Our decision procedure for the Knuth-Bendix ordering constraints is designed as follows. By Lemma 4.2.1 we can transform any constraint into an equivalent disjunction of chained constraints. Our next step is to give a transformation of any chained constraint into an equivalent disjunction of constraints in isolated form. Then in Section 4.3 we show how to transform any constraint in isolated form into an equivalent disjunction of systems of linear Diophantine inequalities on the weights of variables. Then we can use the result that the decidability of systems of linear Diophantine inequalities is in NP.

Let us show how to transform any chained constraint into an equivalent disjunction of isolated forms. The transformation will work on the constraints of the form

$$C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}, \tag{4.2}$$

such that

1. $C_{arith}, C_{triang}, C_{simp}$ are as in the definition of isolated form;

2. $C_{chain}$ is a chained constraint;

3. each variable of $C_{chain}$ neither occurs in $C_{simp}$ nor is dependent in $C_{triang}$.

We will call such constraints (4.2) *working*. Let us call the *size* of a chained constraint $C$ the total number of occurrences of function symbols and variables in $C$. Likewise, the *essential size* of a working constraint is the size of its chained part $C_{chain}$.

At each transformation step we will replace working constraint (4.2) by a disjunction of working constraints but of smaller essential sizes. Evidently, when the essential size is 0, we obtain a constraint in isolated form.

Let us prove some lemmas about solutions to constraints of the form (4.2). Note that any chained constraint is of the form

$$\begin{array}{c} t_{11} \# t_{12} \# \ldots \# t_{1m_1} \\ \succ_w \\ \cdots \\ \succ_w \\ t_{k1} \# t_{k2} \# \ldots \# t_{km_k}, \end{array} \tag{4.3}$$

where each $\#$ is either $=_{\mathrm{TA}}$ or $\succ_{lex}$ and each $t_{ij}$ is a flat term. We call a *row* in such a constraint any maximal subsequence $t_{i1} \# t_{i2} \# \ldots \# t_{im_i}$ in which $\succ_w$ does not occur. So constraint (4.3) contains $k$ rows, the first one is $t_{11} \# t_{12} \# \ldots \# t_{1m_1}$ and the last one $t_{k1} \# t_{k2} \# \ldots \# t_{km_k}$. Note that for any solution to (4.3) all terms in a row have the same weight.

LEMMA 4.2.2 *There exists a polynomial-time algorithm which transforms any chained constraint $C$ into an equivalent chained constraint $C'$ such that the size of $C'$ is not greater than the size of $C$, either $C'$ is $\perp$ or of the form (4.3), and $C'$ has the following property. Suppose some term of the first row $t_{1j}$ of $C'$ is a variable $y$. Then either*

1. *$y$ has exactly one occurrence in $C'$, namely $t_{1j}$ itself; or*
2. *$y$ has exactly two occurrences in $C'$, both in the first row: some $t_{1n}$ has the form $f(y)$ for $n < j$, and $w(f) = 0$; moreover in this case there exists at least one $\succ_{lex}$ between $t_{1n}$ and $t_{1j}$.*

PROOF. Note that if $y$ occurs in any term $t(y)$ which is not in the first row, then $C$ is unsatisfiable, since for any solution $\theta$ to $C$ we have $|y\theta| > |t(y)\theta|$, which is impossible. Suppose that $y$ has another occurrence in a term $t_{1n}$ of the first row. Consider two cases.

1. *$t_{1n}$ coincides with $y$.* Then either $C$ has no solution, or part of the first row between $t_{1n}$ and $t_{1j}$ has the form $y =_{\mathrm{TA}} \ldots =_{\mathrm{TA}} y$. In the latter case part $y =_{\mathrm{TA}}$ can be removed from the first row, so we can assume that no term in the first row except $t_{1j}$ is $y$.

2. $t_{1n}$ *is a nonvariable term containing* $y$. Since $t_{1n}$ and $y$ are in the same row, for every solution $\theta$ to $C$ we have $|y\theta| = |t_{1n}\theta|$. Since $t_{1n}$ is a flat term, by Lemma 3.3.9 the equality $|y\theta| = |t_{1n}\theta|$ is possible only if $t_{1n}$ is $f(y)$, $n < j$ and there exists at least one $\succ_{lex}$ between $t_{1n}$ and $t_{1j}$. Finally, if $f(y)$ has more than one occurrence in the first row, we can get rid of all of them but one in the same way as we got rid of multiple occurrences of $y$.

Note that the transformation presented in this proof can be made in polynomial time. It is also not hard to argue that the transformation does not increase the size of the constraint. □

We will now take a working constraint $C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$, whose chained part satisfies Lemma 4.2.2 and transform it into an equivalent disjunction of working constraints of smaller essential sizes in Lemma 4.2.5 below. More precisely, these constraints will be equivalent when the signature contains no unary function symbol of the weight 0. When the signature contains such a symbol $f$, a weaker notion of equivalence will hold, see formula (4.1) on page 42.

A term $s$ is called an *$f$-variant* of a term $t$ if $s$ can be obtained from $t$ by a sequence of operations of the following forms: replacement of a subterm $f(r)$ by $r$ or replacement of a subterm $r$ by $f(r)$. Evidently, $f$-variant is an equivalence relation. Two substitutions $\theta_1$ and $\theta_2$ are said to be $f$-variants if for every variable $x$ the term $x\theta_1$ is an $f$-variant of $x\theta_2$. In the proof of several lemmas below we will replace a constraint $C(\bar{x})$ by a formula $A(\bar{x}, \bar{y})$ containing extra variables $\bar{y}$ and say that $C(\bar{x})$ and $A(\bar{x}, \bar{y})$ are *equivalent up to $f$*. By this we mean the following.

1. For every substitution $\theta_1$ grounding for $\bar{x}$ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$, there exists a substitution $\theta_2$ grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, and the restriction of $\theta_2$ to $\bar{x}$ is an $f$-variant of $\theta_1$.

2. For every substitution $\theta_2$ grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, there exists a substitution $\theta_1$ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$ and $\theta_1$ is an $f$-variant of the restriction of $\theta_2$ to $\bar{x}$.

In other words, formula (4.1) on page 42 holds. Note that when the signature contains no unary function symbol of the weight 0, equivalence up to $f$ is the same as equality of terms in $\mathrm{TA}^+(\Sigma)$.

LEMMA 4.2.3 *Let $C = C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$ be a working constraint and $\theta_1$ be a solution to $C$. Let $\theta_2$ be an $f$-variant of $\theta_1$ such that*

1. *$\theta_2$ is a solution to $C_{chain}$ and*

2. *$\theta_2$ coincides with $\theta_1$ on all variables not occurring in $C_{chain}$.*

*Then there exists an $f$-variant $\theta_3$ of $\theta_2$ such that*

1. *$\theta_3$ is a solution to $C$ and*

2. *$\theta_3$ coincides with $\theta_2$ on all variables except for the dependent variables of $C_{triang}$.*

PROOF. Let us first prove that $\theta_2$ is a solution to both $C_{arith}$ and $C_{simp}$. Since $C_{simp}$ and $C_{chain}$ have no common variables, it follows that $\theta_1$ and $\theta_2$ agree on all variables of $C_{simp}$, and so $\theta_2$ is a solution to $C_{simp}$. Since $\theta_1$ and $\theta_2$ are $f$-variants and the weight of $f$ is 0, for every term $t$ we have $|t\theta_1| = |t\theta_2|$, whenever $t\theta_1$ is ground. Therefore, $\theta_2$ is a solution to $C_{arith}$ if and only if so is $\theta_1$. So $\theta_2$ is a solution to $C_{arith}$.

It is fairly easy to see that $\theta_2$ can be changed on the dependent variables of $C_{triang}$ obtaining a solution $\theta_3$ to $C$ which satisfies the conditions of the lemma. $\square$

This lemma will be used below in the following way. Instead of considering the set $\Theta_1$ of all solutions to $C_{chain}$ we can restrict ourselves to a subset $\Theta_2$ of $\Theta_1$ as soon as for every solution $\theta_1 \in \Theta_1$ there exists a solution $\theta_2 \in \Theta_2$ such that $\theta_2$ is an $f$-variant of $\theta_1$.

Let us call an *$f$-term* any term of the form $f(t)$. By the *$f$-height* of a term $t$ we mean the number $n$ such that $t = f^n(s)$ and $s$ is not an $f$-term. Note that the $f$-terms are exactly the terms of a positive $f$-height. We call the *$f$-distance* between two terms $s$ and $t$ the difference between the $f$-height of $s$ and $f$-height of $t$. For example, the $f$-distance between the terms $f(a)$ and $f(f(g(a, b)))$ is $-1$.

Let us now prove a lemma which implies that any solution to $C$ can be transformed into a solution with a "small" $f$-height.

LEMMA 4.2.4 *Let $C_{chain}$ be a chained constraint of the form*

$$p_l \# p_{l-1} \# \dots \# p_1 \succ_w \dots,$$

*where each $\#$ is either $=_{\mathrm{TA}}$ or $\succ_{lex}$. Further, let $C_{chain}$ satisfy the conditions of Lemma 4.2.2 and $\theta$ be a solution to $C_{chain}$. Then there exists an $f$-variant $\theta'$ of $\theta$ such that*

1. *$\theta'$ is a solution to $C_{chain}$ and*

2. *for every $k \in \{1, \ldots, l\}$, the $f$-height of $p_k\theta'$ is at most $k$.*

PROOF. Let us first prove the following statement

(4.4) The row $p_l\#p_{l-1}\#\ldots\#p_1$ has a solution $\theta_1$, such that (i) $\theta_1$ is an $f$-variant of $\theta$, (ii) for every $1 < k \leq l$ the $f$-distance between $p_k\theta_1$ and $p_{k-1}\theta_1$ is at most 1.

Suppose that for some $k$ the $f$-distance between $p_k\theta$ and $p_{k-1}\theta$ is $d > 1$. Evidently, to prove (4.4) it is enough to show the following.

(4.5) There exists a solution $\theta_2$ such that (i) $\theta_2$ is an $f$-variant of $\theta$, (ii) the $f$-distance between $p_k\theta_2$ and $p_{k-1}\theta_2$ is $d-1$, and (iii) for every $k' \neq k$ the $f$-distance between $p_{k'}\theta_2$ and $p_{k'-1}\theta_2$ coincides with the $f$-distance between $p_{k'}\theta$ and $p_{k'-1}\theta$.

Let us show (4.5), and hence (4.4). Since $\theta$ is a solution to the row, then for every $k''' \geq k$ the $f$-distance between any $p_{k'''}\theta$ and $p_k\theta$ is nonnegative. Likewise, for every $k'' < k-1$ the $f$-distance between any $p_{k-1}\theta$ and $p_{k''}\theta$ is nonnegative. Therefore, for all $k''' \geq k > k''$, the $f$-distance between $p_{k'''}\theta$ and $p_{k''}\theta$ is $\geq d$, and hence is at least 2. Let us prove the following.

(4.6) Every variable $x$ occurring in $p_l\#p_{l-1}\#\ldots\#p_k$ does not occur in $p_{k-1}\#\ldots\#p_1$.

Let $x$ occur in terms $p_i$ and $p_j$ such that $l \geq i \geq k$ and $k-1 \geq j \geq 1$. Since the constraint satisfies Lemma 4.2.2, then $p_i = f(x)$ and $p_j = x$. Then the $f$-distance between $p_i\theta$ and $p_j\theta$ is 1, but by our assumption it is at least 2, so we obtain a contradiction. Hence (4.6) is proved.

Now note the following.

(4.7) If for some $k''' \geq k$ a variable $x$ occurs in $p_{k'''}$, then $x\theta$ is an $f$-term.

Suppose, by contradiction, that $x\theta$ is not an $f$-term. Note that $p_{k'''}\theta$ has a positive $f$-height, so $p_{k'''}$ is either $x$ of $f(x)$. But we proved before that the $f$-distance between $p_{k'''}\theta$ and $p_{k-1}\theta$ is at least 2, so $x$ must be an $f$-term.

Now, to satisfy (4.5), define the substitution $\theta_2$ as follows:

$$\theta_2(x) = \begin{cases} \theta(x), & \text{if } x \text{ does not occur in } p_l, \dots, p_k; \\ t, & \text{if } x \text{ occurs in } p_l, \dots, p_k \text{ and } \theta(x) = f(t). \end{cases}$$

By (4.6) and (4.7), $\theta_2$ is defined correctly. We claim that $\theta_2$ satisfies (4.5). The properties (i)–(iii) of (4.5) are straightforward by our construction, it only remains to prove that $\theta_2$ is a solution to the row, i.e. for every $k'$ we have $p_{k'}\theta_2 \# p_{k'-1}\theta_2$. Consider the case when $k' > k$. Since $\theta$ is a solution to the row, for each $k'' \geq k$ we have $p_{k''}\theta$ is an $f$-term and hence $p_{k''}$ is either a variable or a term $f(x)$ for some variable $x$. Therefore, by definition of $\theta_2$ we have $p_{k'}\theta = f(p_{k'}\theta_2)$ and $p_{k'-1}\theta = f(p_{k'-1}\theta_2)$, so $p_{k'}\theta_2 \# p_{k'-1}\theta_2$ follows from $p_{k'}\theta \# p_{k'-1}\theta$. When $k' < k$ we have $p_{k'}\theta = p_{k'}\theta_2$ and $p_{k'-1}\theta = p_{k'-1}\theta_2$, hence $p_{k'}\theta_2 \# p_{k'-1}\theta_2$. The only remaining case is $k = k'$.

Assume $k = k'$. Since the $f$-distance between $p_k\theta$ and $p_{k-1}\theta$ is $d > 1$, we have $p_k\theta \neq p_{k-1}\theta$, and hence $p_k \# p_{k-1}$ must be $p_k \succ_{lex} p_{k-1}$. Since $\theta$ is a solution to $p_k \succ_{lex} p_{k-1}$ and since $\theta_2$ is an $f$-variant of $\theta$, the weights of $p_k\theta_2$ and $p_{k-1}\theta_2$ coincide. But then $p_k\theta_2 \succ_{lex} p_{k-1}\theta_2$ follows from the fact that the $f$-distance between $p_k\theta_2$ and $p_{k-1}\theta_2$ is $d - 1 \geq 1$.

Now the proof of (4.5), and hence of (4.4), is completed. In the same way as (4.4), we can also prove

(4.8) The constraint $C_{chain}$ has a solution $\theta'$ such that (i) $\theta'$ is an $f$-variant of $\theta$, (ii) for every $1 < k \leq l$ the $f$-distance between $p_k\theta_1$ and $p_{k-1}\theta'$ is at most 1. (iii) the $f$-height of $p_1\theta'$ is at most 1; (iv) $\theta'$ and $\theta$ coincide on all variables occurring in the rows below the first one.

It is easy to see that $\theta'$ from (4.8) satisfies all conditions required by our lemma. □

The following lemma is the main lemma of this section.

LEMMA 4.2.5 *Let $C = C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$ be a working constraint in which $C_{chain}$ is nonempty. There exists a nondeterministic polynomial-time algorithm which transforms $C$ into a disjunction of working constraints having $C_{chain}$ of smaller sizes and equivalent to $C$ up to $f$.*

PROOF. The proof is rather complex, so we will give a plan of it. The proof is presented as a series of transformations on the first row of $C_{chain}$. These transformations may result in new constraints added to $C_{arith}$, $C_{triang}$, and $C_{simp}$. First, we will get rid of equations $s =_{\mathrm{TA}} t$ in the first row, by introducing *quasi-flat* terms, i.e. terms $f^k(t)$, where $t$ is flat. If the first row contained no function symbols, then we will replace the first row by new constraints added to $C_{simp}$ and $C_{arith}$, thus decreasing the size of the chained part. If there were function symbols in the first row, we will continue as follows.

We will "guess" the values of some variables $x$ of the first row, i.e. replace them by some quasi-flat term $f^m(g(\bar{y}))$, where $\bar{y}$ is a sequence of new variables. After these steps, the size of the first row can, in general, increase. Then we will show how to replace the first row by new constraints involving only variables occurring in the row, but not function symbols. Finally, we will prove that the number of variables from the new constraints that remain in the chained part is not greater than the original number of variables in the first row, and therefore the size of the chained part decreases.

Formally, consider the first row of $C_{chain}$. Let this row be $p_l \# p_{l-1} \# \ldots \# p_1$. Then $C_{chain}$ has the form $p_l \# p_{l-1} \# \ldots \# p_1 \succ_w t_1 \# \ldots \# t_n$. If $l = 1$, i.e., the first row consists of one term, we can remove this row and add $|p_1| > |t_1|$ to $C_{arith}$ obtaining an equivalent constraint with smaller essential size, that is, the size of $C_{chain}$. So we assume that the first row contains at least two terms.

As before, we assume that $f$ is a unary function symbol of the weight 0. By Lemma 4.2.4, if some $p_i$ is either a variable $x$ or a term $f(x)$, it is enough to search for solutions $\theta$ such that the height of $x\theta$ is at most $l$.

A term is called *quasi-flat* if it has the form $f^k(t)$ where $t$ is flat. We will now get rid of equalities in the first row, but by introducing quasi-flat terms instead of the flat ones. When we use notation $f^k(t)$ below, we assume $k \geq 0$, and $f^0(t)$ will stand for $t$. We eliminate equalities from the first row in two steps. First we will eliminate equalities among variables and $f$-terms transforming them into an equivalent set of equalities in triangle form, then we eliminate all other equalities in the first row.

Consider the set $S$ of all equalities $t =_{\mathrm{TA}} s$ occurring in the first row of $C_{chain}$, where $s$ and $t$ are either variables or flat $f$-terms. We will transform $S$ into an equivalent system $F$ in triangle form such that all terms in $F$ will be flat. We assume that before the transformation $F$ is empty. First we replace all equalities in $S$ of the form $f(x) =_{\mathrm{TA}} f(y)$ by $x =_{\mathrm{TA}} y$ obtaining an equivalent system $S'$ in which all equalities are of the form $x =_{\mathrm{TA}} t$. Now, either $S'$ is unsatisfiable or there exists an equality $x =_{\rm TA} t$ in $S'$, such that $x$ does not occur in $f$-terms of $S'$. We move such an equality $x =_{\rm TA} t$ into $F$ and replace all occurrences of $x$ in $S'$ by $t$, obtaining $S''$. It is easy to see that the system $F \cup S''$ is equivalent to $S$, all terms in $F \cup S''$ are flat, $F$ is in triangle form and the number of variables occurring into $S''$ is less than the number of variables occurring into $S$. Repeating this process we can eliminate all variables from $S$ and obtain the required $F$ in polynomial time.

Now we remove from $C_{chain}$ all equalities occurring in $S$. Let us note that variables of $F$ can occur in $C_{chain}$ only in the first row, and only in the terms $f^r(y)$ for $0 \leq r \leq 1$. Next we repeatedly replace all occurrences of dependent variables of $F$ occurring in $C_{chain}$ obtaining an equivalent constraint in chained form with terms of the form $f^k(x)$ where $k$ is bounded by the size of $F$. Finally we move $F$ into $C_{triang}$.

After all these transformations we can assume that equalities $f^k(x) =_{\rm TA} f^m(y)$ do not occur in the first row.

If the first row contains an equality $x =_{\rm TA} t$ between a variable and a term, we replace this equality by $t$, replace all occurrences of $x$ by $t$ in the first row, and add $x =_{\rm TA} t$ to $C_{triang}$ obtaining an equivalent working constraint. Since $x$ can occur only in the terms of the form $f^r(x)$, it is easy to see that these replacements can be done in polynomial time.

If the first row contains an equality $g(x_1, \ldots, x_m) =_{\rm TA} h(t_1, \ldots, t_n)$ where $g$ and $h$ are different function symbols, the constraint is unsatisfiable.

If the first row contains an equality $g(x_1, \ldots, x_n) =_{\rm TA} g(y_1, \ldots, y_n)$ we do the following. If the term $g(x_1, \ldots, x_n)$ coincides with $g(y_1, \ldots, y_n)$, replace this equality by $g(x_1, \ldots, x_n)$. Otherwise, find the smallest number $i$ such that $x_i$ is different from $y_i$ and

1. add $y_i =_{\rm TA} x_i$ to $C_{triang}$;

2. replace all occurrences of $y_i$ in $C_{chain}$ by $x_i$.

We apply this transformation repeatedly until all equalities $g(x_1, \ldots, x_n) =_{\rm TA} g(y_1, \ldots, y_n)$ disappear from the first row.

So we can now assume that the first row contains no equalities and hence it has the form $q_n \succ_{lex} q_{n-1} \succ_{lex} \ldots \succ_{lex} q_1$, where all of the terms $q_i$ are quasi-flat.

If all of the $q_i$ are variables, we can move $q_n \succ_{lex} q_{n-1} \succ_{lex} \ldots \succ_{lex} q_1$ to $C_{simp}$ and add $|q_1| > |t_1|$ to $C_{arith}$ obtaining an equivalent working constraint of smaller essential size. Hence, we can assume that at least one of the $q_i$ is a nonvariable term.

Take any term $q_k$ in the first row such that $q_k$ is either a variable $x$ or a term $f^r(x)$. Note that other occurrences of $x$ in $C_{chain}$ can only be in the first row, and only in the terms of the form $f^k(x)$.

Consider the formula $G$ defined as

$$\bigvee_{g \in \Sigma - \{f\}} \bigvee_{m=0\ldots l} x =_{\mathrm{TA}} f^m(g(\bar{y})). \tag{4.9}$$

where $\bar{y}$ is a sequence of pairwise different new variables. Since we proved that it is enough to restrict ourselves to solutions $\theta$ for which the height of $x\theta$ is at most $l$, the formulas $C$ and $C \wedge G$ are equivalent up to $f$.

Using the distributivity laws, $C \wedge G$ can be turned into an equivalent disjunction of formulas $x =_{\mathrm{TA}} f^m(g(\bar{y})) \wedge C$. For every such formula, replace $x$ by $f^m(g(\bar{y}))$ in the first row, and add $x =_{\mathrm{TA}} f^m(g(\bar{y}))$ to the triangle part. We do this transformation for all terms in the first row of the form $f^k(z)$, where $k \geq 0$ and $z$ is a variable. Now all the terms in the first row are of the form $f^m(g(\bar{y}))$, where $g$ is different from $f$ and $m \geq 0$.

Let us show how to replace constraints of the first row with equivalent constraints consisting of constraints on variables and arithmetical constraints. Consider the pair $q_n, q_{n-1}$. Now $q_n = f^k(g(x_1, \ldots, x_u))$ and $q_{n-1} = f^m(h(y_1, \ldots, y_v))$ for some variables $x_1, \ldots, x_u, y_1, \ldots, y_v$ and function symbols $g, h \in \Sigma - \{f\}$. Then $q_n \succ_{lex} q_{n-1}$ is $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$. If $k < m$ or ($k = m$ and $h \gg g$), then $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$ is equivalent to $\bot$. If $k > m$ or ($k = m$ and $g \gg h$), then $f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v))$ is equivalent to the arithmetical constraint $|g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)|$ which can be added to $C_{arith}$. If $k = m$ and $g = h$ (and hence $u = v$), then

$$\begin{aligned} f^k(g(x_1, \ldots, x_u)) \succ_{lex} f^m(h(y_1, \ldots, y_v)) \leftrightarrow {} & |g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)| \wedge \\ & \bigvee_{i=1\ldots u} (x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ_{KBO} y_i). \end{aligned}$$

We can now do the following. Add $|g(x_1, \ldots, x_u)| =_{\mathbb{N}} |h(y_1, \ldots, y_v)|$ to $C_{arith}$ and replace $q_n \succ_{lex} q_{n-1}$ with the equivalent disjunction

$$\bigvee_{i=1\ldots u} (x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ_{KBO} y_i).$$

Then using the distributivity laws turn this formula into the equivalent disjunction of constraints of the form $C \wedge x_1 =_{\mathrm{TA}} y_1 \wedge \ldots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ_{KBO} y_i$ for all $i = 1 \ldots u$. For each of these constraints, we can move, as before, the equalities $x =_{\mathrm{TA}} y$ one by one to the triangle part $C_{triang}$, and make $C_{chain} \wedge x_i \succ_{KBO} y_i$ into a disjunction of chained constraints as in Lemma 4.2.1.

Let us analyze what we have achieved. After these transformations, in each member of the obtained disjunction the first row is removed from the chained part $C_{chain}$ of $C$. Since the row contained at least one function symbol, each member of the disjunction will contain at least one occurrence of a function symbol less than the original constraint. This is enough to prove termination of our algorithm, but not enough to present it as a nondeterministic polynomial-time algorithm. The problem is that, when $p_n$ is a variable $x$ or a term $f(x)$, one occurrence of $x$ in $p_n$ can be replaced by one or more constraints of the form $x_i \succ_{KBO} y_i$, where $x_i$ and $y_i$ are new variables. To be able to show that the essential sizes of each of the resulting constraints is strictly less than the essential size of the original constraint, we have to modify our algorithm slightly.

The modification will guarantee that the number of new variables introduced in the chained part of the constraint is not greater than the number of variables eliminated from the first row. We will achieve this by moving some constraints to the simple part $C_{simp}$. The new variables only appear when we replace a variable in the first row by a term $f^k(h(u_1, \ldots, u_m))$ or by $f^k(h(v_1, \ldots, v_m))$ obtaining a constraint $f^k(h(u_1, \ldots, u_m)) \succ_{lex} f^k(h(v_1, \ldots, v_m))$, which is then replaced by

$$u_1 =_{\mathrm{TA}} v_1 \wedge \ldots \wedge u_{i-1} =_{\mathrm{TA}} v_{i-1} \ \wedge u_i \succ_{KBO} v_i. \tag{4.10}$$

Let us call a variable $u_i$ (respectively, $v_i$) *new* if $f^k(h(u_1, \ldots, u_m))$ occurred in the terms of the first row when we replaced a variable by a nonvariable term containing $h$ using formula (4.9). In other words, new variables are those that did not occur in the terms of the first row before our transformation, but appeared in the terms of the first row during the transformation. All other variables are called *old*. After the transformation we obtain a conjunction $E$ of constraints of the form $x_i =_{\mathrm{TA}} x_j$ or $x_i \succ_{KBO} x_j$, where $x_i, x_j$ can be either new or old. Without loss of generality we can assume that this conjunction of constraints does not contain chains of the form $x_1 \# \ldots \# x_n \# x_1$ where $n \geq 2$ and at least one of the $\#$'s is $\succ_{KBO}$. Indeed, if $E$ contains such a chain, then it is unsatisfiable.

We will now show that the number of new variables can be restricted by moving constraints on these variables into the triangle or simple part. Among the new variables, let us distinguish the following three kinds of variables. A new variable $x$ is called *blue in* $E$ if $E$ contains a chain $x =_{\mathrm{TA}} x_1 =_{\mathrm{TA}} \ldots =_{\mathrm{TA}} x_n$, where $x_n$ is an old variable. Evidently, a blue variable $x$ causes no harm since it can be replaced by an old variable $x_n$. Let us denote by $\prec$ the inverse relation to $\succ_{KBO}$. A new variable $x$ is called *red in* $E$ if it is not blue in $E$ and $E$ contains a chain $x\#x_1\#\ldots\#x_n$, where $x_n$ is an old variable, and all of the #'s are either $=_{\mathrm{TA}}$, or $\succ_{KBO}$, or $\prec$. Red variables are troublesome, since there is no obvious way to get rid of them. However, we will show that the number of red variables is not greater than the number of replaced variables (such as the variable $x$ in (4.9)). Finally, all new variables that are neither blue nor red in $E$ are called *green* in $E$.

**Getting rid of the green variables.** We will now show that the green variables can be moved to the simple part of the constraint $C_{simp}$. To this end, note an obvious property: if $E$ contains a constraint $x\#y$ and $x$ is green, then $y$ is green too. We can now do the following with the green variables. As in Lemma 4.2.1, we can turn all the green variables into a disjunction of chained constraints of the form $v_1\#\ldots\#v_n$, where # are $=_{\mathrm{TA}}$, $\succ_w$, or $\succ_{lex}$, and use the distributivity laws to obtain chained constraints $v_1\#\ldots\#v_n$. Let us call this constraint a *green chain*. Then, if there is any equality $v_i =_{\mathrm{TA}} v_{i+1}$ in the green chain, we add this equality to $C_{triang}$ and replace this equality by $v_{i+1}$ in the chain. Further, if the chain has the form $v_1 \succ_{lex} \ldots \succ_{lex} v_k \succ_w v_{k+1}\#\ldots\#v_n$, we add $v_1 \succ_{lex} \ldots \succ_{lex} v_k$ to $C_{simp}$ and $|v_k| > |v_{k+1}|$ to $C_{arith}$, and replace the green chain by $v_{k+1}\#\ldots\#v_n$. We do this transformation until the green chain becomes of the form $v_1 \succ_{lex} \ldots \succ_{lex} v_k$. After this, the green chain can be removed from $E$ and added to $C_{simp}$. Evidently, this transformation can be presented as a nondeterministic polynomial-time algorithm.

**The red variables.** Let us show the following: in every term $f^k(h(u_1,\ldots,u_m))$ in the first row at most one variable among $u_1,\ldots,u_m$ is red. It is not hard to argue that it is sufficient to prove a stronger statement: if for some $i$ the variable $u_i$ is red or blue, then all variables $u_1,\ldots,u_{i-1}$ are blue. So suppose that $u_i$ is either red or blue and $u_i\#y_n\#\ldots\#y_1$ is a shortest chain in $E$ such that $y_1$ is old. We prove that the variables $u_1,\ldots,u_{i-1}$ are blue, by induction on $n$. When $n = 1$ and $u_i$ is red, $E$ contains either $u_i \succ_{KBO} y_1$ or $y_1 \succ_{KBO} u_i$, where $y_1$ is old. Without loss of generality assume that $E$ contains $u_i \succ_{KBO} y_1$. Then (cf. (4.10)) this equation appeared in $E$ when we replaced $f^k(h(u_1, \ldots, u_m)) \succ_{lex} f^k(h(v_1, \ldots, v_m))$ by $u_1 =_{\text{TA}} v_1 \wedge \ldots \wedge u_{i-1} =_{\text{TA}} v_{i-1} \ \wedge u_i \succ_{KBO} v_i$ and $y_1 = v_i$. But then $E$ also contains the equations $u_1 =_{\text{TA}} v_1, \ldots, u_{i-1} =_{\text{TA}} v_{i-1}$, where the variables $v_1, \ldots, v_{i-1}$ are old, and so the variables $u_1, \ldots, u_{i-1}$ are blue. In the same way we can prove that if $u_i$ is blue then $u_1, \ldots, u_{i-1}$ are blue. The proof for $n > 1$ is similar, but we use the fact that $v_1, \ldots, v_{i-1}$ are blue rather than old.

To complete the transformation, we add all constraints on the red and the old variables to $C_{chain}$ and make $C_{chain}$ into a disjunction of chained constraints as in Lemma 4.2.1.

**Getting rid of the blue variables.** If $E$ contains a blue variable $x$, then it also contains a chain of constraints $x =_{\text{TA}} x_1 =_{\text{TA}} \ldots =_{\text{TA}} x_n$, where $x_n$ is an old variable. We replace $x$ by $x_n$ in $C$ and add $x =_{\text{TA}} x_n$ to the triangle part $C_{triang}$.

When we completed the transformation on the first row, the row disappears from the chained part $C_{chain}$ of $C$. If the first row contained no function symbols, the size of $C_{chain}$ will become smaller, since several variables will be removed from it. If $C_{chain}$ contained at least one function symbol, then after the transformation the number of occurrences of function symbols in $C_{chain}$ will decrease. Some red variables will be introduced, but we proved that their number is not greater than the number of variables eliminated from the first row. Therefore, the size of $C_{chain}$ strictly decreases after the transformation due to elimination of at least one function symbol.

Again, it is not hard to argue that the transformation can be presented as a nondeterministic polynomial-time algorithm computing all members of the resulting disjunction of constraints.

$\square$

Lemmas 4.2.1 and 4.2.5 imply the following:

LEMMA 4.2.6 *Let $C$ be a constraint. Then there exists a disjunction $C_1 \vee \ldots \vee C_n$ of constraints in isolated form equivalent to $C$ up to $f$. Moreover, members of such a disjunction can be found by a nondeterministic polynomial-time algorithm.*
$\square$

Our next aim is to present a nondeterministic polynomial-time algorithm solving constraints in isolated form.

## 4.3 From constraints in isolated form to systems of linear Diophantine inequalities

Let $C$ be a constraint in isolated form

$$C_{simp} \wedge C_{arith} \wedge C_{triang}.$$

Our decision algorithm will be based on a transformation of the simple constraint $C_{simp}$ into an equivalent disjunction $D$ of arithmetical constraints. Then, in Section 4.4 we show how to check the satisfiability of the resulting formula $D \wedge C_{arith} \wedge C_{triang}$ by using an algorithm for solving systems of linear Diophantine inequalities on the weights of variables.

To transform $C_{simp}$ into an arithmetical formula, observe the following. The constraint $C_{simp}$ is a conjunction of the constraints of the form

$$x_1 \succ_{lex} \cdots \succ_{lex} x_N$$

having no common variables. To solve such a constraint we have to ensure that there exist at least $N$ different terms of the same weight as $x_1$ (since the Knuth-Bendix order is total).

In this section we will show that for each $N$ the statement "there exists at least $N$ different terms of a weight $w$" can be expressed in the Presburger Arithmetic as an existential formula of one variable $w$.

We say that a relation $R(\bar{x})$ on natural numbers is $\exists$-*definable*, if there exists an existential formula of Presburger Arithmetic $C(\bar{x}, \bar{y})$ such that $R(\bar{x})$ is equivalent to $\exists \bar{y} C(\bar{x}, \bar{y})$. We call a function $r(\bar{x})$ $\exists$-definable if so is the relation $r(\bar{x}) = y$. Note that composition of $\exists$-definable functions is $\exists$-definable.

Let us fix an enumeration $g_1, \ldots, g_S$ of the signature $\Sigma$. We assume that the first $B$ symbols $g_1, \ldots, g_B$ is a sequence of all symbols in $\Sigma$ of arity $\geq 2$, and the first $F$ symbols $g_1, \ldots, g_F$ is a sequence all nonconstant symbols in $\Sigma$. The arity of each $g_i$ is denoted by $arity_i$. *In this section we assume that $B$, $F$, $S$, and the weight function $w$ are fixed.*

We call the *contents* of a ground term $t$ the tuple of natural numbers $(n_1, \ldots, n_S)$ such that $n_i$ is the number of occurrences of $g_i$ in $t$ for all $i$. For example, if the sequence of elements of $\Sigma$ is $g, h, a, b$, and $t = h(g(h(h(a)), g(b, b)))$, the contents of $t$ is $(2, 3, 1, 2)$.

LEMMA 4.3.1 *The following relation exists*$(x, n_1, \ldots, n_S)$ *is* $\exists$*-definable: there exists at least one ground term of* $\Sigma$ *of the weight* $x$ *and contents* $(n_1, \ldots, n_S)$.

PROOF. We will define *exists*$(x, n_1, \ldots, n_S)$ by a conjunction of two linear Diophantine inequalities.

The first equation is

$$x = \sum_{1 \le i \le S} w(g_i) \cdot n_i. \tag{4.11}$$

It is not hard to argue that this equation says: every term with the contents $(n_1, \ldots, n_S)$ has weight $x$.

The second formula says that the number of constant and nonconstant function symbols in $(n_1, \ldots, n_S)$ is appropriately balanced for constructing a term:

$$1 + \sum_{1 \le i \le S} (arity_i - 1) \cdot n_i = 0. \tag{4.12}$$

□

Let us prove some lower bounds on the number of terms of a fixed weight.

We leave the following two lemmas to the reader. The first one implies that, if there exists any ground term $t$ of a weight $x$ with at least $N$ occurrences of nonconstant symbols, including at least one occurrence of a function symbol of an arity $\ge 2$, then there exists at least $N$ different ground terms of the weight $x$.

LEMMA 4.3.2 *Let* $x, n_1, \ldots, n_S$ *be natural numbers such that exists*$(x, n_1, \ldots, n_S)$ *holds,* $n_1 + \ldots + n_B \ge 1$ *and* $n_1 + \ldots + n_F \ge N$*. Then there exist at least* $N$ *different ground terms with the contents* $(n_1, \ldots, n_S)$. □

The second lemma implies that, if there exists any ground term $t$ of a weight $x$ with at least $N$ occurrences of nonconstant function symbols, including at least two different unary function symbols, then there exists at least $N$ different ground terms of the weight $x$.

LEMMA 4.3.3 *Let* $x, n_1, \ldots, n_s$ *be natural numbers such that exists*$(x, n_1, \ldots, n_S)$ *holds,* $n_1 + \ldots + n_F \ge N$ *and at least two numbers among* $n_{B+1}, \ldots, n_F$ *are positive. Then there exists at least* $N$ *different ground terms with the contents* $(n_1, \ldots, n_S)$.
□

Let us note that if our signature consists only of a unary function symbol of a positive weight and constants, then the number of different terms in any weight is less or equal to the number of constants in the signature.

The remaining types of signatures are covered by the following lemma.

LEMMA 4.3.4 *Let $\Sigma$ contain a function symbol of an arity greater than or equal to 2, or contain at least two different unary function symbols. Then there exist two natural numbers $N_1$ and $N_2$ such that for all natural numbers $N$ and $x$ such that $x > N \cdot N_1 + N_2$, the number of terms of the weight $x$ is either 0 or greater than $N$.*

PROOF. If $\Sigma$ contains a unary function symbol of the weight 0 then the number of different terms of any weight is either 0 or $\omega$ and the lemma trivially holds.

Therefore we can assume that our signature contains no unary function symbol of the weight 0. Define

$$\begin{array}{rcl} W & = & \max\{w(g_i) | 1 \leq i \leq S\}; \\ A & = & \max\{arity_i | 1 \leq i \leq S\}; \\ N_1 & = & W \cdot A; \\ N_2 & = & W^2 \cdot (A+1) + W. \end{array}$$

Take any $N$ and $x$ such that $x > N \cdot N_1 + N_2$.

Let us prove that if there exists a term of the weight $x$ then the number of occurrences of nonconstant function symbols in this term is greater than $N$. Assume the opposite, i.e. there exists a term $t$ of the weight $x$ such that the number of occurrences of nonconstant function symbols in $t$ is $M \leq N$. Let $(n_1, \ldots, n_S)$ be the contents of $t$ and $L$ denote the number of occurrences of constants in $t$. Note that (4.12) implies $L = 1 + \sum_{1 \leq i \leq F}(arity_i - 1) \cdot n_i$. Then using (4.11) we obtain

$$\begin{array}{l} N \cdot N_1 + N_2 < |t| = \sum_{1 \leq i \leq S} w(g_i) \cdot n_i \leq W \cdot \sum_{1 \leq i \leq S} n_i = \\ W \cdot (M + L) = W \cdot (M + 1 + \sum_{1 \leq i \leq F}(arity_i - 1) \cdot n_i) \leq \\ W \cdot (M + 1 + (A-1) \sum_{1 \leq i \leq F} n_i) = W \cdot (M + 1 + (A-1) \cdot M) = \\ W \cdot (M \cdot A + 1) \leq W \cdot (N \cdot A + 1) < N \cdot N_1 + N_2. \end{array}$$

So we obtain a contradiction.

Consider the following possible cases.

1. *There exists a term of the weight $x$ with an occurrence of a function symbol of an arity greater than or equal to* 2. In this case by Lemma 4.3.2 the number of different terms of the weight $x$ is greater than $N$.

2. *There exists a term of the weight $x$ with occurrences of at least two different unary function symbols.* In this case by Lemma 4.3.3 the number of different terms of the weight $x$ is greater than $N$.

3. *All terms of the weight $x$ have the form $g^k(c)$ for some unary function symbol $g$ and a constant $c$.* We show that this case is impossible. In particular, we show that for any nonconstant function symbol $h$ there exists a term of the weight $x$ in which $g$ and $h$ occur, therefore we obtain a contradiction with the assumption.

   We have $x = w(g) \cdot k + w(c)$. Denote by $H$ the arity of $h$. Let us define integers $M_1, M_2, M_3$ as follows

$$\begin{aligned} M_1 &= w(g); \\ M_2 &= k - w(h) - w(c) \cdot (H-1); \\ M_3 &= w(g)(H-1)+1. \end{aligned}$$

   Let us prove that $M_1, M_2, M_3 > 0$ and there exists a term of the weight $x$ with $M_1$ occurrences of $h$, $M_2$ occurrences of $g$ and $M_3$ occurrences of $c$ and hence obtain a contradiction.

   Since $g$ is unary, $w(g) > 0$, and so $M_1 > 0$. Since $H \geq 1$, we have $M_3 > 0$. Let us show that $M_2 > 0$, i.e. $k > w(h) + w(c) \cdot (H-1)$. We have

$$\begin{aligned} & k = (x - w(c))/w(g) > (N \cdot N_1 + N_2 - w(c))/w(g) \geq \\ & (N_2 - w(c))/w(g) = (W^2 \cdot (A+1) + W - w(c))/w(g) \geq \\ & (W^2 \cdot (A+1))/w(g) \geq W \cdot (A+1) = W + W \cdot A \geq \\ & w(h) + w(c) \cdot A > w(h) + w(c) \cdot (H-1). \end{aligned}$$

   It remains to show that there exists a term of the weight $x$ with $M_1$ occurrences of $h$, $M_2$ occurrences of $g$ and $M_3$ occurrences of $c$. To this end we have to prove (cf. (4.11) and (4.12))

$$x = w(h) \cdot M_1 + w(g) \cdot M_2 + w(c) \cdot M_3;$$
$$1 + (H - 1) \cdot M_1 + (1 - 1) \cdot M_2 + (0 - 1)M_3 = 0.$$

This equalities can be verified directly by replacing $M_1, M_2, M_3$ by their definitions and $x$ by $w(g) \cdot k + w(c)$. □

Define the binary function *tnt* (truncated number of terms) as follows: $tnt(N, M)$ is the minimum of $N$ and the number of terms of the weight $M$ and let us show that *tnt* can be computed in time polynomial of $N+M$. To give a polynomial-time algorithm for this function we need an auxiliary definition and a lemma.

DEFINITION 4.3.5 Let $(n_1, \ldots, n_s)$ and $(m_1, \ldots, m_s)$ be two tuples of natural numbers. We say that $(n_1, \ldots, n_s)$ *extends* $(m_1, \ldots, m_s)$ if $n_i \geq m_i$ for $1 \leq i \leq s$. □

The *depth* of a term is defined by induction as usual: the depth of every constant is 1 and the depth of every nonconstant term $g(t_1, \ldots, t_n)$ is equal to the maximum of the depth of the $t_i$'s plus 1.

LEMMA 4.3.6 *Let $t_1, \ldots, t_n$ be a collection of different terms of the same depth and Con be the contents of a term such that Con extends the contents of all terms $t_i$, $1 \leq i \leq n$. Then there exist at least $n$ different terms with the contents Con.*

PROOF. Let us define the notion of *leftmost subterm* of a term $t$ as follows: every constant $c$ has only one leftmost subterm, namely $c$ itself, and leftmost subterms of a nonconstant term $g(r_1, \ldots, r_n)$ are this term itself and all leftmost subterms of $r_1$. Evidently, for each positive integer $d$ and term $t$, $t$ has at most one leftmost subterm of the depth $d$.

It is not hard to argue that from the condition of the lemma it follows that for every term $t_i$ there exists a term $s_i$ with the contents $Con$ such that $t_i$ is a leftmost subterm of $s_i$. But then the terms $s_1, \ldots, s_n$ are pairwise different, since they have different leftmost subterms of the depth $d$. □

LEMMA 4.3.7 *Let the signature $\Sigma$ contain no unary function symbol of the weight 0 and contain either a function symbol of an arity greater than or equal to 2 or contain at least two different unary function symbols. Then the function $tnt(N, M)$*

*is computable in time polynomial of $M + N$.*

PROOF. It is not hard to argue that for every contents $(n_1, \ldots, n_S)$ such that some of the $n_i$'s is greater than $M$, any term with these contents has the weight greater than $M$. The number of different contents in which each of the $n_i$'s is less than or equal to $M$ is $M^S$, i.e. it is polynomial in $M$, moreover, all these contents can be obtained by an algorithm working in time polynomial in $M$.

Therefore it is sufficient to describe a polynomial-time algorithm which for all contents $(n_1, \ldots, n_S)$, where $1 \leq n_i \leq M$, returns the minimum of $N$ and the number of terms with these contents.

Let us fix contents $Con = (n_1, \ldots n_S)$ where $1 \leq n_i \leq M$. Using equations (4.11) and (4.12), one can check in polynomial time whether there exists a term with the contents $Con$, so we assume that there exists at least one such term.

Our algorithm constructs, step by step, sets $T_0, T_1, \ldots$, of different terms with contents which can be extended to the contents $Con$. Each set $T_i$ will consist only of terms of the depth $i$.

1. *Step 0.* Define $T_0 = \emptyset$.

2. *Step $i + 1$.* Define

$$\begin{aligned} T_{i+1} = \{g(t_1, \ldots, t_m) \mid \ & g \in \Sigma,\ t_1, \ldots, t_m \in T_1 \cup \ldots \cup T_i, \\ & Con \text{ extends the content of } g(t_1, \ldots, t_m), \text{ and} \\ & \text{the depth of } g(t_1, \ldots, t_m) \text{ is } i + 1\}. \end{aligned}$$

   If $T_{i+1}$ has $N$ or more terms, then by Lemma 4.3.6 there exists at least $N$ different terms of the content $Con$, so we terminate and return $N$. If $T_{i+1}$ is empty, we return as the result the minimum of $N$ and the number of terms with the content $Con$ in $T_1 \cup \ldots \cup T_{i+1}$.

Let us prove some obvious properties of this algorithm.

1. *If some $T_i$ contains $N$ or more terms, then there exists at least $N$ terms with the content $Con$.* As we noted, this follows from Lemma 4.3.6.

2. *At the end of step $i + 1$ the set $T_1 \cup \ldots \cup T_{i+1}$ contains all the terms with the contents $Con$ of the depth $\leq i + 1$.* This property obviously holds by our construction.

This property ensures that the algorithm is correct. To prove that it works in time polynomial in $M + N$ it is enough to note that each step can be made in time polynomial in $N$ and the total number of steps is at most $M + 1$. $\square$

Now we are ready to prove the main lemma of this section.

LEMMA 4.3.8 *There exists a polynomial time of $N$ algorithm, which constructs an existential formula $at\_least_N(x)$ valid on a natural number $x$ if and only if there exists at least $N$ different terms of the weight $x$.*

PROOF. If the signature $\Sigma$ contains a unary function symbol of the weight 0 then the number of different terms in any weight is either 0 or $\omega$. Therefore we can define $at\_least_N(x)$ as $\exists n_1 \ldots \exists n_S exists(x, n_1, \ldots, n_S)$.

Let us consider the case when the signature $\Sigma$ consists of a unary function symbol $g$ of a positive weight and constants. For every constant $c$ in $\Sigma$ consider the formula $G_c(x) = \exists k(w(g)k + w(c) = x)$. It is not hard to argue that $G_c(x)$ holds if and only if there exists a term of the form $g^k(c)$ of weight $x$. Let $P$ be the set of all sets of cardinality $N$ consisting of constants of $\Sigma$ (the cardinality of $P$ is obviously polynomial in $N$). It is easy to see that

$$at\_least_N(x) \leftrightarrow \bigvee_{Q \in P} \bigwedge_{c \in Q} G_c(x).$$

It remains to consider the case when our signature contains a function symbol of an arity greater than or equal to 2, or contains at least two different unary function symbols. By Lemma 4.3.4, there exist constants $N_1$ and $N_2$ such that for any natural number $x$ such that $x > N \cdot N_1 + N_2$ the number of terms of the weight $x$ is either 0 or greater than $N$. Let us denote $N \cdot N_1 + N_2$ as $M$ and the set $\{M' | M' \leq M \wedge tnt(N, M') \geq N\}$ as $W$. By Lemmas 4.3.4, 4.3.7 we have

$$at\_least_N(x) \leftrightarrow (\exists n_1, \ldots, n_S exists(x, n_1, \ldots, n_S) \ \wedge x > M) \vee (\bigvee_{M' \in W} x = M').$$

$\square$

## 4.4 Main results

In this section we complete the proofs of the main results of this chapter.

THEOREM 4.4.1 *For every Knuth-Bendix order, the problem of solving ordering constraints is contained in NP.*

PROOF. Take a constraint. By Lemma 4.2.5 it can be effectively transformed into an equivalent disjunction of isolated forms, so it remains to show how to check satisfiability of constraints in isolated form.

Suppose that $C$ is a constraint in isolated form. Recall that $C$ is of the form

$$C_{arith} \wedge C_{triang} \wedge C_{simp}. \tag{4.13}$$

Let $C_{simp}$ contain a chain $x_1 \succ_{lex} \ldots \succ_{lex} x_N$ such that $x_1, \ldots, x_N$ does not occur in the rest of $C_{simp}$. Denote by $C'_{simp}$ the constraint obtained from $C_{simp}$ by removing this chain. It is easy to see that $C$ is equivalent to the constraint

$$C_{arith} \wedge C_{triang} \wedge C'_{simp} \wedge \bigwedge_{i=2\ldots N} (|x_i| =_{\mathbb{N}} |x_1|) \wedge at\_least_N(|x_1|).$$

In this way we can replace $C_{simp}$ by an arithmetical constraint, so we assume that $C_{simp}$ is empty. Let $C_{triang}$ have the form

$$y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_n =_{\mathrm{TA}} t_n.$$

Let $Z$ be the set of all variables occurring in $C_{arith} \wedge C_{triang}$. It is not hard to argue that $C_{arith} \wedge C_{triang}$ is satisfiable if and only if the following constraint is satisfiable:

$$C_{arith} \wedge |y_1| =_{\mathbb{N}} |t_1| \wedge \ldots \wedge |y_n| =_{\mathbb{N}} |t_n| \wedge \textstyle\bigwedge_{z \in Z} at\_least_1(|z|).$$

So we reduced the decidability of the existential theory of term algebras with a Knuth-Bendix order to the problem of solvability of systems of linear Diophantine inequalities. Our proof can be represented as a nondeterministic polynomial-time algorithm. $\square$

This theorem implies the main result of this chapter. Let us call a signature $\Sigma$ *trivial* if it consists of one constant symbol. Evidently, the first-order theory of the term algebra of a trivial signature is polynomial.

THEOREM 4.4.2 *The existential first-order theory of any term algebra of a non-trivial signature with the Knuth-Bendix order is NP-complete.*

PROOF. The containment in NP follows from Theorem 4.4.1. NP-hardness is proved in Proposition 3.5.3 by reducing propositional satisfiability to the existential theory of the term algebra (even without the order). □

Let us show that for some Knuth-Bendix orders even constraint solving can be NP-hard.

EXAMPLE 4.4.3 Consider the signature $\Sigma = \{s, g, h, c\}$, where $h$ is binary, $s, g$ are unary, and $c$ is a constant. Define the weight of all symbols as 1, and use any order $\gg$ on $\Sigma$ such that $g \gg s$. Our aim is to represent any linear Diophantine equation by Knuth-Bendix constraints. To this end, we will consider any ground term $t$ as representing the natural number $|t| - 1$.

Define the formula

$$\begin{array}{l} \mathit{equal\_weight}(x, y) \leftrightarrow \\ \qquad g(x) \succ_{KBO} s(y) \wedge g(y) \succ_{KBO} s(x). \end{array}$$

Obviously, for any ground terms $r, t$ $\mathit{equal\_weight}(r, t)$ holds if and only if $|r| = |t|$.

It is enough to consider systems of linear Diophantine equations of the form

$$x_1 + \ldots + x_n + k = x_0, \tag{4.14}$$

where $x_0, \ldots, x_n$ are pairwise different variables, and $k \in \mathbb{N}$. Consider the constraint

$$\begin{array}{l} \mathit{equal\_weight}(s^{k+2}(h(y_1, h(y_2, \ldots, \\ \qquad\qquad h(y_{n-1}, y_n)))), \\ \qquad s^{2n}(y_0)). \end{array} \tag{4.15}$$

It is not hard to argue that

(4.16) Formula (4.15) holds if and only if

$$|y_1| - 1 + \ldots + |y_n| - 1 + k = |y_0| - 1.$$

Using (4.16), we can transform any system $D(x_0, \ldots, x_n)$ of linear Diophantine equations of the form (4.14) into a constraint $C(y_0, \ldots, y_n)$ such that for every tuple of ground terms $t_0, \ldots, t_n$, $C(t_0, \ldots, t_n)$ holds if and only if so does $D(|t_0| - 1, \ldots, |t_n| - 1)$.

Similar, using a formula

$$\begin{array}{r} greater\_weight(x,y) \leftrightarrow \\ s(x) \succ_{KBO} g(y) \end{array}$$

one can represent systems of linear inequalities using Knuth–Bendix constraints.

It is easy to see that this reduction can be done in polynomial time, assuming that coefficients of linear Diophantine equations and inequalities are represented in the unary notation. $\square$

Since it is well-known that solving linear Diophantine equations with coefficients represented in the unary notation is NP-hard, we have the following theorem.

THEOREM 4.4.4 *For some Knuth-Bendix orders, the problem of solving ordering constraints is NP-complete.* $\square$

This result does not hold for all non-trivial signatures, as the following theorem shows.

LEMMA 4.4.5 *There exists a polynomial time algorithm which solves ordering constraints for any given term algebra over a signature consisting of constants and any total ordering $\succ$ on that term algebra.*

PROOF. Let $\Sigma = \{c_1, \ldots, c_n\}$, w.l.o.g. we can assume that $c_n \succ c_{n-1} \succ \ldots \succ c_1$. Let $C$ be an ordering constraint. First we get rid of equalities as follows. If $t =_{\mathrm{TA}} s$ occurs in $C$ and $t$ is syntactically equal to $s$ then we remove $t =_{\mathrm{TA}} s$ from $C$, if $t$ is a variable then we replace all occurrences of $t$ in $C$ by $s$ and remove $t =_{\mathrm{TA}} s$ from $C$, otherwise $t$ and $s$ are different constants and $C$ is unsatisfiable. Now $C$ consists of conjunctions of atomic formulas of the form $t \succ s$. We define a relation $\succ'_C$ on terms as follows: $t \succ'_C s$ if and only if $t \succ s$ occurs in $C$. Let $\succ_C$ denote a transitive closure of $\succ'_C$. It is easy to see, that using a polynomial time algorithm for transitive closure, we can compute the relation $t \succ_C s$ in polynomial time. Note that if $\succ_C$ is not a strict order then the constraint $C$ is unsatisfiable. So we assume that $\succ_C$ is a strict partial order.

Now we replace all variables in $C$ by constants as follows. Take a variable $x$ such that there is no variable less than $x$ w.r.t. $\succ_C$. There are two possible cases:

1. $x$ is a minimal term w.r.t. $\succ_C$, then we replace all occurrences of $x$ in $C$ by $c_1$.

2. there exist some constants less than $x$ w.r.t. $\succ_C$, then let $c_{max}$ be the greatest w.r.t. $\succ$ constant among such constants. If $c_{max}$ is the maximal constant in TA($\Sigma$) then the constraint $C$ is unsatisfiable, otherwise we replace all occurrences of $x$ by $c_{max+1}$.

Repeating this process we replace all variables in $C$ in polynomial time. To complete the proof of the lemma, it remains to show that transformations 1,2 above, preserve satisfiability of constraints without equality. To this end, we consider a constraint $C$ without equality and a solution $\theta$ to $C$. If the transformation 1 is applicable to $C$ then it is easy to see that

$$\theta'(x) = \begin{cases} c_1, \text{ if } x \text{ is a minimal term w.r.t. } \succ_C, \\ \theta(x) \text{ otherwise.} \end{cases}$$

is a solution to the constraint obtained after applying the transformation 1 to $C$.

Similar one can show that the transformation 2 preserves satisfiability of constraints without equality. □

COROLLARY 4.4.6 *There exists a polynomial time algorithm which checks solvability of ordering constraints for any given Knuth–Bendix order on any term algebra over a signature consisting of constants.* □

As we mentioned in Section 4.1, if we consider real-valued Knuth-Bendix orders then even comparison of ground terms might be undecidable. Let us show it on the following example.

EXAMPLE 4.4.7 Consider a non-computable real number $r$ such that $0 < r < 1$, i.e. there is no algorithm which given a positive integer $n$ computes $r$ with the precision $1/n$, in other words finds two natural numbers $p, q$ such that $|r - p/q| < 1/n$.

Now we consider a signature consisting of two unary symbols $g, h$ and a constant $c$ and consider any Knuth–Bendix order $\succ_{KBO}$ on the corresponding term algebra, such that $w(g) = 1$ and $w(h) = r$. Let us show that comparison of terms in this Knuth–Bendix order is undecidable. Consider a positive integer $n$. Then, it is easy to see that there exists a positive integer $m$ such that $g^m(c) \succ_{KBO} h^n(c) \succ_{KBO} g^{m-1}(c)$. Since $|g^m(c)| \neq |h^n(c)| \neq |g^{m-1}(c)|$, we have $|g^m(c)| > |h^n(c)| > |g^{m-1}(c)|$. From the definition of the weight function we have that $m > rn > m - 1$ and therefore $m/n > r > \frac{m-1}{n}$. Let us take $p = m - 1$ and $q = n$, then we have $|r - p/q| < 1/n$. Therefore using comparison of terms we can compute $r$ with the precision $1/n$. This implies that comparison of terms for this Knuth–Bendix order is undecidable. $\square$

# Chapter 5

# First–order Knuth–Bendix ordering constraints for unary signatures

This chapter is based on the paper [Korovin and Voronkov 2002].

## 5.1 Introduction

In resolution-based theorem proving there are important simplifications which allow us to remove clauses from the search space (for example subsumption). It turns out that in order to express applicability conditions for these simplifications, we need to consider constraints which involve first-order quantifiers (see Chapter 2). Unfortunately the first-order theory of the recursive path orders is undecidable [Treinen 1990, Comon and Treinen 1997]. Only recently the decidability of the first-order theory of recursive path orders over unary signatures has been proven [Narendran and Rusinowitch 2000]. A signature is called unary if it consists of unary function symbols and constants.

In this chapter we prove the following result.

**Theorem 5.3.2:** *The first-order theory of any Knuth-Bendix order over any unary signature is decidable.*

Our decision procedure uses interpretation of unary terms as trees and uses decidability of the weak monadic second-order theory of binary trees.

This chapter is structured as follows. In Section 5.2 we introduce the notion of interpretation and show how it can be used to prove decidability of a given theory by reducing this problem to decidability of some known theory. In Section 5.3 we show how to interpret unary terms with any Knuth-Bendix order in the weak monadic second-order theory of binary trees, which decidability is well-known.

In this chapter we will only consider signatures consisting of unary function symbols and constants.

## 5.2 Interpretations

Interpretations play an important role in mathematical logic, allowing us to describe the properties of a given structure based on the properties of another structure.

We will use an interpretation of first-order structures with the Knuth-Bendix order, in the structure of two successors considered in the weak monadic second-order language. The weak monadic second-order language is a language closed under $\vee, \wedge, \neg$, which extends first-order language with variables $X, Y, \ldots$ ranging over finite sets, includes atomic formulas $t \in X$ where $t$ is a first order term and allows quantifiers over the set variables.

Let us introduce a simple notion of interpretation which we will use later to show the decidability of the first-order theory of Knuth-Bendix orders over unary signatures. For a more general theory of interpretations see, e.g., [Hodges 1993, Ershov 1980, Rabin 1977]. In the sequel we will use lower-case letters $x, y, z, \ldots$ to denote first-order variables and upper-case letters $X, Y, Z, \ldots$ to denote second-order variables.

DEFINITION 5.2.1 Let $A$ be a structure in a first-order language $L_A$ and $B$ be a structure in a weak monadic second-order language $L_B$. We say that the structure $A$ is interpretable in the structure $B$ if there exist a positive integer $m$ and the following formulas:

1. $\phi_{domain}(\bar{X})$, where $\bar{X}$ is a tuple of second-order variables of the length $m$ such that the set $A' = \{\bar{S} \mid B \models \phi_{domain}(\bar{S})\}$ is non-empty;

2. $\phi_g(\bar{X}_1, \ldots, \bar{X}_n, \bar{Y})$ for each function symbol $g$ in the language $L_A$, where the arity of $g$ is $n$ and $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of second-order variables of the

length $m$, and this formula defines a function, denoted by $g'$, on $A'$, i.e., we have

$$g'(\bar{S}_1, \ldots, \bar{S}_n) = \bar{T} \Leftrightarrow B \models \phi_g(\bar{S}_1, \ldots, \bar{S}_n, \bar{T});$$

3. $\phi_P(\bar{X}_1, \ldots, \bar{X}_n)$ for each predicate symbol $P$ in $L_A$, where the arity of $P$ is $n$ and $\bar{X}_1, \ldots, \bar{X}_n$ are tuples of second-order variables of the length $m$, and this formula defines a predicate on $A'$, denoted by $P'$, i.e., we have

$$P'(\bar{S}_1, \ldots, \bar{S}_n) \Leftrightarrow B \models \phi_P(\bar{S}_1, \ldots, \bar{S}_n);$$

such that the following condition holds.
The structure with the domain $A'$, in which every function symbol $f$ is interpreted by the function $f'$ and every predicate symbol $P$ is interpreted by $P'$, is isomorphic to the structure $A$. □

We will use the following fundamental property of interpretability.

PROPOSITION 5.2.2 *If a structure $A$ is interpretable in the structure $B$ and the theory of $B$ (in the language $L_B$) is decidable, then the theory of $A$ (in the language $L_A$) is also decidable.* □

The proof can be found, e.g. in [Hodges 1993, Ershov 1980, Rabin 1977].

## 5.3 Interpretation of the Knuth-Bendix order in WS2S

We will use interpretations to show the decidability of the first-order theory of Knuth-Bendix orders over unary signatures. We show how to interpret Knuth-Bendix orders in the structure of two successors in the weak monadic language. Then, using the result [Thatcher and Wright 1968] on the decidability of the weak monadic theory of two successors, we conclude that the first-order theory of Knuth-Bendix orders over unary signatures is decidable.

Let us briefly recall the definition of the structure of two successors (see, e.g., [Comon, Dauchet, Gilleron, Jacquemard, Lugiez, Tison and Tommasi 1997] for details). The domain consists of finite binary strings including the empty string

$\lambda$. There are two functions $0(x)$ and $1(x)$ which add 0 and 1 respectively to the end of the string. For example $0(101) = 1010$. Instead of $0(t)$ and $1(t)$ we will write, respectively, $t \cdot 0$ and $t \cdot 1$. The atomic formulas are equalities $t = s$ between first-order terms, and $t \in X$ where $t$ is a first-order term. Formulas are built from atomic formulas using logical connectives $\wedge, \vee, \neg$, the first-order quantifiers $\exists x, \forall x$ and second-order quantifiers over finite sets $\exists X, \forall X$. We will use the following standard shorthands: $\exists x \in X \phi(x, X)$ for $\exists x(x \in X \wedge \phi(x, X))$ and $\forall x \in X \phi(x, X)$ for $\forall x(x \in X \supset \phi(x, X))$. Binary strings can be seen as positions in binary trees, and in the sequel we sometimes will use the word *position* instead of string.

Below we will use the following definable relations on sets with a straightforward meaning.

**Emptiness:**

$$X = \emptyset \leftrightarrow \forall x(x \notin X).$$

**Intersection:**

$$X \cap Y = Z \leftrightarrow \forall x(x \in Z \leftrightarrow (x \in X \wedge x \in Y)).$$

**Union:**

$$X \cup Y = Z \leftrightarrow \forall x(x \in Z \leftrightarrow (x \in X \vee x \in Y)).$$

**Partition:**

$$\mathit{Partition}(X, X_1, \ldots, X_n) \leftrightarrow X = \bigcup_{1 \leq i \leq n} X_i \wedge \bigwedge_{1 \leq i < j \leq n} X_i \cap X_j = \emptyset.$$

**PrefixClosed:**

$$\mathit{PrefixClosed}(X) \leftrightarrow \forall x((x \cdot 0 \in X \vee x \cdot 1 \in X) \supset x \in X).$$

Sets satisfying *PrefixClosed* will be called *trees.*

**Prefix order $\sqsubseteq$:**

$$x \sqsubseteq y \leftrightarrow \forall X((y \in X \wedge \mathit{PrefixClosed}(X)) \supset x \in X).$$

Likewise, we introduce

$$x \sqsubset y \leftrightarrow x \sqsubseteq y \wedge x \neq y.$$

**Lexicographic order $\leq_{lex}$:**

$$x \leq_{lex} y \leftrightarrow x \sqsubseteq y \bigvee \exists z(z \cdot 0 \sqsubseteq x \wedge z \cdot 1 \sqsubseteq y).$$

Likewise, we introduce

$$x <_{lex} y \leftrightarrow x \leq_{lex} y \wedge x \neq y.$$

**Maximal prefix:** Informally, $MaxPref(m, X)$ says that $m$ is a maximal element in $X$ w.r.t. the prefix order.

$$MaxPref(m, X) \leftrightarrow m \in X \wedge \forall z \in X \neg(m \sqsubset z).$$

**Minimal prefix:** Informally, $MinPref(m, X)$ says that $m$ is a minimal element in $X$ w.r.t. the prefix order.

$$MinPref(m, X) \leftrightarrow m \in X \wedge \forall z \in X \neg(z \sqsubset m).$$

**Maximal lexicographically:** Informally, $MaxLex(m, X)$ says that $m$ is a maximal element in $X$ w.r.t. the lexicographic order.

$$MaxLex(m, X) \leftrightarrow m \in X \wedge \forall z \in X \neg(m <_{lex} z).$$

Assuming a fixed Knuth-Bendix order we will show how to interpret it in the structure of two successors using the weak monadic second-order language.

Let us consider a signature $\Sigma = \{g_1, \ldots, g_s\}$ consisting of unary function symbols and constants. From now on we assume that $\Sigma$ is fixed and denote by $s$ the number of function symbols and constants in it. We denote the set of constants in $\Sigma$ by $\Sigma_c$ and the set of unary function symbols by $\Sigma_g$. Let $w$ be a weight function on $\Sigma$ and $\gg$ be a precedence relation compatible with $w$. Also $f$ will always denote the function symbol of weight zero. Denote the Knuth-Bendix order induced by this weight function and precedence relation by $\succ$. Now we show how to interpret $\mathrm{TA}_{\succ}(\Sigma)$ in the structure of two successors using the weak monadic language.

We define the interpretation in three steps. First we map terms into labelled trees and define functions and relations on them such that the obtained structure will be isomorphic to $\mathrm{TA}_{\succ}(\Sigma)$. Then we show how labelled trees can be represented as $s+1$-tuples of finite sets of binary strings. Finally we show how to define these representations, and corresponding functions and relations on them in the structure of two successor using weak monadic second–order logic.

**Coding of terms.**

The labelled trees are binary trees labelled with the function symbols. We want tree representation of terms to satisfy the following properties

1. The functions of $\mathrm{TA}_{\succ}(\Sigma)$ can be defined in the monadic second-order language.
2. The function symbols of the term algebra are represented in such a way that we can compare weights of terms using the monadic second-order language.
3. For the terms of equal weight we should be able to compare their top function symbols and then lexicographically compare their subterms.

Let us start with an example. Consider a signature $\{f(), g(), h(), c\}$, and a weight function $w$ such that $w(f) = 0, w(g) = 2, w(h) = w(c) = 1$. Figure 5.1 shows how to construct a labelled tree representing the term $f(h(f(f(g(c)))))$. The labelled tree is built by traversing the tree inside-out, for example, the root of the labelled tree is labelled with the constant $c$. We would like the rightmost branch of the tree to have the length equal to the weight of the term. To this end, we repeat every function symbol of a positive weight the number of times equal to its weight. Since the function symbol $f$ has the weight 0, it is not included on the rightmost branch. To represent this symbol, we make branching to the left at the corresponding points of the tree.

Before giving a formal definition of the representation of terms as labelled trees, let us consider trees as sets of binary strings. Any binary tree without labels can be defined as a set of binary strings, namely the positions of the nodes in the tree. For example, the tree of Figure 5.1 contains the binary strings $\lambda$

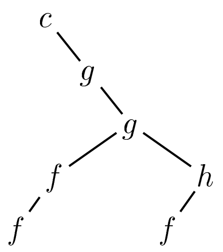


Figure 5.1: The labelled tree representation of $fhffgc$, $w(f) = 0$, $w(g) = 2$, $w(h) = w(c) = 1$

labelled with $c$, strings 1 and 11 labelled as $g$, string 111 labelled by $h$, and strings 110, 1100, and 1110 labelled by $f$.

Formally, for each term $t$ we define a labelled binary tree $\mathit{Tree}_t$ and two positions $\mathit{Right}_t$ and $\mathit{Top}_t$ in this tree. The definition is by induction on $t$.

1. If $t$ is a constant $c$ of a weight $w$, then $\mathit{Tree}_t$ consists of the strings $\lambda, 1, \ldots, 1^{w-1}$, labelled by $c$, and $\mathit{Right}_t = \mathit{Top}_t = 1^{w-1}$.
2. If $t = f(t')$, then $\mathit{Tree}_t$ is obtained from $\mathit{Tree}_{t'}$ by adding the string $\mathit{Top}_{t'} \cdot 0$ labelled by $f$, and we have $\mathit{Top}_t = \mathit{Top}_{t'} \cdot 0$, $\mathit{Right}_t = \mathit{Right}_{t'}$.
3. If $t = g(t')$, where $g$ has a positive weight $w$, then $\mathit{Tree}_t$ is obtained from $\mathit{Tree}_{t'}$ by adding the strings $\mathit{Right}_{t'} \cdot 1, \ldots, \mathit{Right}_{t'} \cdot 1^w$ labelled by $g$, and we have $\mathit{Top}_t = \mathit{Right}_t = \mathit{Top}_{t'} \cdot 1^w$.

The mapping $t \mapsto \mathit{Tree}_t$ defines the embedding of terms into labelled trees.

Now it is easy to define the functions of the term algebra $\mathrm{TA}_{\succ}(\Sigma)$ on the labelled trees. We define the value of a function $g$ on the labelled tree representation of a term $t$ to be equal to the labelled tree representation of the term $g(t)$. Likewise, we can define the Knuth-Bendix order on such trees. It is evident that the obtained structure on the labelled trees is isomorphic to $\mathrm{TA}_{\succ}(\Sigma)$.

Now we will show how to represent labelled trees by $s+1$-tuples. Let $T$ be a labelled tree whose set of positions is $X$. Then we represent $T$ as the tuple $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$, where each set $X_{g_i}$ is the set of positions labelled by $g_i$ and $X$ is the set of all positions in this tree. If a term $t$ is represented by a labelled tree $T$, and $T$ is represented by a tuple $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$, we will also say that the tuple $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$ represents the term $t$.

To complete our construction, we have to show how to define in the second-order monadic language the set of tuples which represent the terms of $\mathrm{TA}_{\succ}(\Sigma)$, and then show that all functions and predicates of $\mathrm{TA}_{\succ}(\Sigma)$ are definable on the representation.

To this end we introduce some auxiliary definable predicates on sets of strings.

**OneSucc:** Informally, $OneSucc(X)$ says that the set of strings $X$ consists of strings of 1's, contains the empty string, and is prefix closed.

$$OneSucc(X) \leftrightarrow \lambda \in X \wedge (\forall x \in X(x \neq \lambda \supset \exists y \in X\ x = y \cdot 1\ )).$$

**Spine:** The set of strings on rightmost branch of a tree will be called the *spine* of this tree. $Spine(X, Y)$ says that $X$ is a tree and $Y$ is its spine.

$$\begin{aligned} Spine(X,Y) \quad \leftrightarrow \quad & PrefixClosed(X) \wedge OneSucc(Y) \wedge Y \subseteq X \wedge \\ & \forall Y'((Y' \subseteq X \wedge OneSucc(Y')) \supset Y' \subseteq Y). \end{aligned}$$

**Comb:** Informally, $Comb(X)$ says that $X$ is a tree and all right-branching positions in it are in its spine.

$$\begin{aligned} Comb(X) \quad \leftrightarrow \quad & PrefixClosed(X) \wedge \\ & \forall x(x \cdot 1 \in X \supset \exists Y\, Spine(X,Y) \wedge x \in Y). \end{aligned}$$

**LabelledTree:** Informally, $LabelledTree(X, X_{g_1}, \ldots, X_{g_s})$ says that $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$ is a tuple which is a labelled tree (not necessarily representing a term) appropriately labelled in the following sense: all positions along its spine are labelled with function symbols of positive weights and all other positions are labelled with the function symbol of the weight 0.

$$\begin{aligned} LabelledTree(X, X_{g_1}, \ldots, X_{g_s}) \quad \leftrightarrow \quad & Partition(X, X_{g_1}, \ldots, X_{g_s}) \\ & \wedge\ Comb(X) \\ & \wedge\ Spine(X, \cup_{g \in \Sigma \setminus \{f\}} X_g). \end{aligned}$$

The labelled trees defined by $LabelledTree(X, X_{g_1}, \ldots, X_{g_s})$ are similar to those representing terms, except that in our representation of terms each occurrence of a function symbol of a positive weight should be repeated the number of times equal to the weight. Let us express this restriction in the weak monadic second-order logic.

A set consisting of strings of 1's will be called a *1-set*. A 1-set which is a set of successive positions we be called an *interval*. The *length* of an interval is the number of elements in it. Consider a labelled tree $\langle X, X_{g_1} \ldots, X_{g_s} \rangle$ and a function symbol $g \in \Sigma \backslash \{f\}$. First we introduce notions of $g$-interval and maximal $g$-interval. A *$g$-interval* is an interval which is contained in $X_g$ and contains no branching positions with a possible exception of the maximal position of this interval.

**$g$-interval:** Let $g \in \Sigma \setminus \{f\}$. Informally $\mathit{Interval}_g(I, \bar{X})$ says that $\bar{X}$ is a labelled tree and $I$ is a $g$-interval.

$$\begin{array}{rcl} \mathit{Interval}_g(I, \bar{X}) & \leftrightarrow & \mathit{LabelledTree}(\bar{X}) \wedge I \subseteq X_g \wedge \\ & & \exists m_0, m_1 (\mathit{MinPref}(m_0, I) \wedge \mathit{MaxPref}(m_1, I) \\ & & \qquad \wedge \forall y (m_0 \sqsubseteq y \sqsubseteq m_1 \supset y \in I)) \\ & & \wedge \forall z \in I( \neg \mathit{MaxPref}(z, I) \supset z \cdot 0 \notin X). \end{array}$$

**Maximal $g$-interval:** is a $g$-interval that can not be properly extended.

$$\mathit{MaxInterval}_g(I, \bar{X}) \leftrightarrow \mathit{Interval}_g(I, \bar{X}) \wedge \forall J (\mathit{Interval}_g(J, \bar{X}) \supset I \not\subset J).$$

Our next goal is to express that the length of every maximal $g$-interval is a multiple of $w(g)$. To this end we introduce a notion of $n$-interval, for each positive $n$. We say that a position $x$ is the $n$-successor of a position $y$ if $x = y \cdot 1^n$. An *$n$-interval* is a 1-set which consists of a sequence of positions such that each next position is an $n$-successor of the previous. We always assume that an $n$-interval contains at least two elements. For example, the following set is a 2-interval $\{1, 111, 11111\}$. Let us show that for a given $n$, the property of being an $n$-interval is expressible in the monadic second-order logic.

**1-set:**

$$\mathit{OneSet}(X) \leftrightarrow \exists Y X \subseteq Y \wedge \mathit{OneSucc}(Y).$$

**$n$-interval:**

$$
\begin{aligned}
\mathit{Interval}_n(X) \;\leftrightarrow\; & \mathit{OneSet}(X) \wedge \exists m(\mathit{MinPref}(m, X) \wedge 1^n(m) \in X) \\
& \wedge \forall y \in X(\mathit{MaxPref}(y, X) \vee (y \cdot 1^n \in X \wedge \textstyle\bigwedge_{1 \le i < n} y \cdot 1^i \notin X)).
\end{aligned}
$$

Now, to say that the length of every maximal $g$-interval in a tree is a multiple of $w(g)$, it is enough to say that for every maximal $g$-interval in the tree, its minimal point and the successor of its maximal point are in some $w(g)$-interval.

**Preterm:** Informally, $\mathit{Preterm}(\bar{X})$ says that $\bar{X}$ is a labelled tree and the length of every maximal $g$-interval in this tree is a multiple of $w(g)$.

$$
\begin{aligned}
\mathit{Preterm}(\bar{X}) \;\leftrightarrow\; & \mathit{LabelledTree}(\bar{X}) \wedge \\
& \textstyle\bigwedge_{g \in \Sigma \setminus \{f\}} \forall I(\mathit{MaxInterval}_g(I, \bar{X}) \supset \\
& \qquad \exists m_0 \exists m_1 (\mathit{MinPref}(m_0, I) \wedge \mathit{MaxPref}(m_1, I) \wedge \\
& \qquad\quad \exists Y\, \mathit{Interval}_{w(g)}(Y) \wedge m_0 \in Y \wedge m_1 \cdot 1 \in Y)).
\end{aligned}
$$

Finally, to define terms we need to say that the root position of a term is a constant and there are no other occurrences of constants.

**Term:**

$$
\begin{aligned}
\mathit{Term}(\bar{X}) \;\leftrightarrow\; & \mathit{Preterm}(\bar{X}) \wedge \lambda \in \textstyle\bigcup_{g \in \Sigma_c} \wedge \\
& \textstyle\bigwedge_{g \in \Sigma_c} (X_g \neq \emptyset \supset \lambda \in X_g \wedge \mathit{MaxPref}(1^{(w(g)-1)}(\lambda), X_g)).
\end{aligned}
$$

So, we have that $\mathit{Term}(\bar{X})$ defines the domain of our term algebra in the structure of two successors. Let us now show how to define the functions of the term algebra and the Knuth-Bendix order on this domain. Each constant can be easily defined as following.

**Constants:** For each constant $c \in \Sigma_c$ define

$$
\phi_c(\bar{X}) \leftrightarrow \mathit{Term}(\bar{X}) \wedge X_c = \cup_{0 \le i < w(c)} \{1^i(\lambda)\} \wedge X = X_c.
$$

Now we consider a function symbol $g \in \Sigma_g \setminus \{f\}$. In order to say that $\bar{Y} = g(\bar{X})$ we need to say that the spine of $\bar{Y}$ extends the spine of $\bar{X}$ with $g$ repeated $w(g)$ times.

**Function symbols of positive weight:** For each function symbol $g \in \Sigma_g \setminus \{f\}$ define

$$\begin{array}{rcl}\phi_g(\bar{X},\bar{Y}) & \leftrightarrow & \mathit{Term}(\bar{X}) \wedge \mathit{Term}(\bar{Y}) \wedge \bigwedge_{h \in \Sigma \setminus \{g\}} X_h = Y_h \wedge \\ & & \exists S \exists m (\mathit{Spine}(X,S) \wedge \mathit{MaxLex}(m,S) \wedge \\ & & \quad Y_g = (X_g \cup \bigcup_{1 \le i \le w(g)} \{1^i(m)\})).\end{array}$$

In order to say that $\bar{Y} = f(\bar{X})$ where $f$ is the function symbol of zero weight we need to say that $\bar{Y}$ extends the greatest position in $\bar{X}$, w.r.t. lexicographic order, with $f$.

**Function symbol of zero weight:** For the function symbol of zero weight define

$$\begin{array}{rcl}\phi_f(\bar{X},\bar{Y}) & \leftrightarrow & \mathit{Term}(\bar{X}) \wedge \mathit{Term}(\bar{Y}) \wedge \bigwedge_{h \in \Sigma \setminus \{f\}} X_h = Y_h \wedge \\ & & \exists m (\mathit{MaxLex}(m,X) \wedge Y_f = (X_f \cup \{m \cdot 0\})).\end{array}$$

Finally, we will define the Knuth-Bendix order. For this we need some auxiliary predicates.

**Point of difference:** Informally, $\mathit{PointOfDifference}(x,\bar{X},\bar{Y})$ says that $\bar{X},\bar{Y}$ represent terms and they differ at the position $x$.

$$\begin{array}{rl}\mathit{PointOfDifference}(x,\bar{X},\bar{Y}) & \leftrightarrow \mathit{Term}(\bar{X}) \wedge \mathit{Term}(\bar{Y}) \wedge \\ & \bigvee_{g \in \Sigma} ((x \in X_g \wedge x \notin Y_g) \vee (x \in Y_g \wedge x \notin X_g)).\end{array}$$

**Maximal point of difference:** Informally, $\mathit{MaxPointOfDifference}(x,\bar{X},\bar{Y})$ says that $\bar{X},\bar{Y}$ are terms, and $x$ is the greatest point of difference w.r.t. the lexicographic order.

$$\begin{array}{rl}\mathit{MaxPointOfDifference}(x,\bar{X},\bar{Y}) & \leftrightarrow \mathit{PointOfDifference}(x,\bar{X},\bar{Y}) \wedge \\ & \forall y (\mathit{PointOfDifference}(y,\bar{X},\bar{Y}) \supset y \le_{lex} x).\end{array}$$

Now we are ready to define the Knuth-Bendix order. Indeed, to say that $\bar{X} \succ \bar{Y}$ it is enough to say that $\bar{X},\bar{Y}$ are terms, the maximal point of their difference is in $X$ and the function symbol at this position in $\bar{X}$ is greater in the precedence relation $\gg$ than the function symbol at this position in $\bar{Y}$, if this position belongs to $Y$.

**Knuth-Bendix order:**

$$\begin{aligned}\bar{X} \succ \bar{Y} \quad \leftrightarrow \quad & \exists x(\mathit{MaxPointOfDifference}(x, \bar{X}, \bar{Y}) \wedge x \in X \wedge \\ & \textstyle\bigwedge_{g \in \Sigma}(x \in X_g \supset (x \notin Y \vee \bigvee_{h \ll g} x \in Y_h))).\end{aligned}$$

Lemma 5.3.1 *The formulas $\mathit{Term}(\bar{X}), \bar{X} \succ \bar{Y}$ and $\phi_g(\bar{X}, \bar{Y})$ for $g \in \Sigma$, define an interpretation of the term algebra with the Knuth-Bendix order in the structure of two successors.*

Proof. The claim follows from the definition of the Knuth-Bendix order. □

Using the decidability of the weak monadic second-order theory of two successors, this lemma and Proposition 5.2.2 we obtain the main result of this chapter.

Theorem 5.3.2 *The first-order theory of any Knuth-Bendix order over any unary signature is decidable.* □

Let us note that this interpretation of Knuth-Bendix orders also works if we consider partial precedence order $\ll$ on the signature, assuming that $f$ is the greatest symbol w.r.t. $\ll$.

Finally, let us remark that our result can be easily extended to the decidability of term algebras with several Knuth-Bendix orders which have the same weight functions and different precedence relations. Indeed, in this case the interpretation of terms and term functions is the same as above and we only need to add formulas $\bar{X} \succ_i \bar{Y}$ for each Knuth-Bendix order $\succ_i$.

# Chapter 6

# Orientability of rewrite rules by Knuth-Bendix orders

This chapter is based on papers [Korovin and Voronkov 2001*b*, Korovin and Voronkov 2003*d*].

Let us give an informal overview of the results proved in this chapter. The formal definitions will be given in the next section. Let $\succ$ be any order on ground terms and $l \to r$ be a rewrite rule. We say that $\succ$ *orients* $l \to r$, if for every ground instance $l' \to r'$ of $l \to r$ we have $l' \succ r'$. We write $l \succeq r$ if for every ground instance $l' \to r'$ of $l \to r$ we have $l' \succ r'$ or $l' = r'$. There are situations where we want to check if there *exists* a simplification order on ground terms that orients a given system of (possibly non-ground) rewrite rules. We call this problem *orientability*. Orientability can be useful when a theorem prover is run on a new problem for which no suitable simplification order is known, or when termination of a rewrite system is to be established automatically (see Chapter 2).

We give a polynomial-time algorithm for checking orientability by Knuth-Bendix orders.

**Theorem 6.9.1:** *The problem of the existence of a Knuth-Bendix order which orients a given term rewriting system can be solved in polynomial time.*

The main algorithmic complexity of our orientability algorithm arises from the usage of solvability of homogeneous linear inequalities. We show that this is unavoidable by reducing solvability of certain homogeneous linear inequalities to our orientability problem. Using this reduction and a reduction to the circuit value problem we show the following hardness result.

> **Theorem 6.9.2:** *The problem of orientability of term rewriting systems by Knuth-Bendix orders is P-complete. Moreover, it is P-hard even for ground rewriting systems.*

A similar problem of orientability by the non-ground version of real-valued Knuth-Bendix orders was studied in [Dick et al. 1990] and an algorithm for orientability was given. We prove that any term rewriting system orientable by a real-valued Knuth-Bendix order is also orientable by an integer-valued Knuth-Bendix order. This result also holds for the non-ground version of Knuth-Bendix orders considered in [Dick et al. 1990]. In our proofs we use some techniques of [Dick et al. 1990]. We also show that some rewrite systems could not be oriented by non-ground version of Knuth-Bendix orders, but can be oriented by our algorithm.

The second problem we consider is solving ordering constraints consisting of a single inequality, over a given Knuth-Bendix order. If $\succ$ is total on ground terms, then the problem of checking if $\succ$ orients $l \to r$ has relation to the problem of solving ordering constraints over $\succ$. Indeed, $\succ$ *does not* orient $l \to r$ if and only if there exists a ground instance $l' \to r'$ of $l \to r$ such that $r' \succeq l'$, i.e., if and only if the ordering constraint $r \succeq l$ has a solution. This means that any procedure for solving ordering constraints consisting of a single inequality can be used for checking whether a given system of rewrite rules is oriented by $\succ$, and vice versa. Using the same technique as for the orientability problem, we show that the problem of solving Knuth-Bendix ordering constraints consisting of single inequalities can be solved in polynomial time. Let us remark that this algorithm does not use solvability of systems of homogeneous linear inequalities and runs in the time $O(n^2)$ of the size of the constraint.

> **Theorem 6.9.3:** *The problem of solving a given Knuth-Bendix ordering constraint consisting of a single inequality can be solved in the time $O(n^2)$.*

## 6.1 Preliminaries

In the sequel we will often refer to the least and the greatest terms among the terms of the minimal weight for a given Knuth-Bendix order. It is easy to see that every term of the minimal weight is either a constant of the minimal weight, or a term $f^n(c)$, where $c$ is a constant of the minimal weight, and $w(f) = 0$. Therefore, the least term of the minimal weight is always the constant of the minimal weight which is the least among all such constants w.r.t. $\gg$. This constant is also the least term w.r.t. $\succ$.

The greatest term of the minimal weight exists if and only if there is no unary function symbol of the weight 0. In this case, this term is the constant of the minimal weight which is the greatest among such constants w.r.t. $\gg$.

DEFINITION 6.1.1 (grounding substitution) A substitution $\theta$ is *grounding* for an expression $E$ (i.e., term, rewrite rule etc.) if for every variable $x$ occurring in $E$ the term $\theta(x)$ is ground. We denote by $E\theta$ the expression obtained from $E$ by replacing in it every variable $x$ by $\theta(x)$. A *ground instance* of an expression $E$ is any expression $E\theta$ which is ground. □

A *rewrite rule* is a pair of terms $(l, r)$, possibly with variables, usually denoted by $l \to r$. A *term rewriting system* is a finite set of term rewrite rules. The following definition is central to this chapter.

DEFINITION 6.1.2 (orientability) A Knuth-Bendix order $\succ$ *orients* a rewrite rule $l \to r$ if for every ground instance $l' \to r'$ of $l \to r$ we have $l' \succ r'$. A Knuth-Bendix order *orients a system* $R$ of rewrite rules if it orients every rewrite rule in $R$. □

We show that the problem of the existence of a Knuth-Bendix order which orients a given system of term rewrite rules can be solved in polynomial time. Moreover, if the given system of rewrite rules is orientable by a Knuth-Bendix order, we can find such an order in polynomial time.

The decidability of the orientability problem for Knuth-Bendix orders does not follow immediately from the decidability of Knuth-Bendix ordering constraints (Chapter 4), as it is in the case of recursive path orders. For a given finite signature, there exists only a finite number of different recursive path orders. But there exists an infinite number of different Knuth-Bendix orders, since the weight function is not restricted.

We define orientability in terms of ground instances of rewrite rules. One can also define orientability using the non-ground version of Knuth-Bendix orders as originally defined by Knuth and Bendix [1970]. But then we obtain a weaker notion (fewer systems can be oriented) as the following example shows.

EXAMPLE 6.1.3 Consider the following rewrite rule:

$$g(x, a, b) \to g(b, b, a). \tag{6.1}$$

For any choice of the weight function $w$ and order $\gg$, $g(x, a, b) \succ_{KBO} g(b, b, a)$ does not hold for the original Knuth-Bendix order with variables. However, rewrite rule (6.1) can be oriented by any Knuth-Bendix order such that $w(a) \geq w(b)$ and $a \gg b$. □

In fact the order based on all ground instances is the greatest simplification order extending the Knuth-Bendix order from ground terms to non-ground terms.

## 6.2 Systems of homogeneous linear inequalities

In our proofs and in the algorithm we will use several properties of homogeneous linear inequalities. The definitions related to systems of linear inequalities can be found in standard textbooks, see, e.g., Schrijver [1998]. We will denote column vectors of variables by $X$, integer or real vectors by $V, W$, integer or real matrices by $A, B$. Column vectors consisting of 0's will be denoted by $\mathbf{0}$. The set of real numbers is denoted by $\mathbb{R}$, and the set of non-negative real numbers by $\mathbb{R}^+$.

DEFINITION 6.2.1 (homogeneous linear inequalities) A *homogeneous linear inequality* has the form either $VX \geq 0$ or $VX > 0$. A *system of homogeneous linear inequalities* is a finite set of homogeneous linear inequalities. □

Solutions (real or integer) to systems of homogeneous linear inequalities are defined as usual. When we write a system of homogeneous linear inequalities as $AX \geq \mathbf{0}$, we assume that every inequality in the system is of the form $VX \geq 0$ (but not of the form $VX > 0$).

We will use the following fundamental property of system of homogeneous linear inequalities:

LEMMA 6.2.2 *Let $AX \geq \mathbf{0}$ be a system of homogeneous linear inequalities, where $A$ is an integer matrix. Then there exists a finite number of integer vectors $V_1, \ldots, V_n$ such that the set of solutions to $AX \geq \mathbf{0}$ is*

$$\{r_1V_1 + \ldots + r_nV_n \mid r_1, \ldots, r_n \in \mathbb{R}^+\}. \tag{6.2}$$

□

The proof can be found in, e.g., [Schrijver 1998].

The following lemma was proved in [Martin 1987] for the systems of linear homogeneous inequalities over the real numbers. We will give a simpler proof of it here.

LEMMA 6.2.3 *Let $AX \geq \mathbf{0}$ be a system of homogeneous linear inequalities where $A$ is an integer matrix and let* Sol *be the set of all real solutions to the system. Then the system can be split into two disjoint subsystems $BX \geq \mathbf{0}$ and $CX \geq \mathbf{0}$ such that*

1. *$BV = \mathbf{0}$ for every $V \in$ Sol.*
2. *If $C$ is non-empty then there exists a solution $V \in$ Sol such that $CV > \mathbf{0}$.*

PROOF. By Lemma 6.2.2 we can find integer vectors $V_1, \ldots, V_n$ such that the set Sol is (6.2). We define $BX \geq \mathbf{0}$ to be the system consisting of all inequalities $WX \geq 0$ in the system such that $WV_i = 0$ for all $i = 1, \ldots, n$; then property 1 is obvious.

Note that the system $CX \geq \mathbf{0}$ consists of the inequalities $WX \geq 0$ such that for some $i$ we have $WV_i > 0$. Take $V$ to be $V_1 + \ldots + V_n$, then it is not hard to argue that $CV > \mathbf{0}$. □

Let $\mathbb{W}$ be a system of homogeneous linear inequalities. We will call the subsystem $BX \geq \mathbf{0}$ of $\mathbb{W}$ the *degenerate subsystem* if the following holds. Denote by $C$ the matrix of the complement to $BX \geq \mathbf{0}$ in $\mathbb{W}$ and by Sol the set of all real solutions to $\mathbb{W}$. Then

1. $BV = \mathbf{0}$ for every $V \in$ Sol.
2. If $C$ is non-empty then there exists a solution $V \in$ Sol such that $CV > \mathbf{0}$.

For every system $\mathbb{W}$ of homogeneous linear inequalities the degenerate subsystem of $\mathbb{W}$ will be denoted by $\mathbb{W}^=$. Note that the degenerate subsystem is defined for arbitrary systems, not only those of the form $AX \geq 0$.

Let us now prove another key property of integer systems of homogeneous linear inequalities: the existence of a real solution implies the existence of an integer solution.

LEMMA 6.2.4 *Let $\mathbb{W}$ be a system of homogeneous linear inequalities with an integer matrix. Let $V$ be a real solution to this system and for some subsystem of $\mathbb{W}$ with the matrix $C$ we have $CV > \mathbf{0}$. Then there exists an integer solution $V'$ to $\mathbb{W}$ for which we also have $CV' > \mathbf{0}$.*

PROOF. Let $\mathbb{W}'$ be obtained from $\mathbb{W}$ by replacement of all strict equalities $WX > 0$ by their non-strict versions $WX \geq 0$. Take vectors $V_1, \dots, V_n$ so that the set of solutions to $\mathbb{W}'$ is (6.2). Evidently, for every inequality $WX \geq 0$ in $CX \geq \mathbf{0}$ there exists some $i$ such that $WV_i > 0$. Define $V'$ as $V_1 + \dots + V_n$, then it is not hard to argue that $CV' > 0$. We claim that $V'$ is a solution to $\mathbb{W}$. Assume the converse, then there exists an inequality $WX > 0$ in $\mathbb{W}$ such that $WV' = 0$. But $WV' = 0$ implies that $WV_i = 0$ for all $i$, so $\mathbb{W}$ has no real solution, contradiction. □

The following lemma follows from Lemmas 6.2.3 and 6.2.4.

LEMMA 6.2.5 *Let $\mathbb{W}$ be a system of homogeneous linear inequalities with an integer matrix and its degenerate subsystem is different from $\mathbb{W}$. Let $C$ be the matrix of the complement of the degenerate subsystem. Then there exists an integer solution $V$ to $\mathbb{W}$ such that $CV > \mathbf{0}$.* □

The following result is well-known, see, e.g., [Schrijver 1998].

LEMMA 6.2.6 *The existence of a real solution to a system of linear inequalities can be decided in polynomial time.* □

This lemma and Lemma 6.2.4 imply the following key result.

LEMMA 6.2.7 *(i) The existence of an integer solution to an integer system of homogeneous linear inequalities can be decided in polynomial time. (ii) If an integer system $\mathbb{W}$ of homogeneous linear inequalities has a solution, then its degenerate subsystem $\mathbb{W}^=$ can be found in polynomial time.*

PROOF. (i) By Lemma 6.2.6 the existence of a real solution can be checked in polynomial time. By Lemma 6.2.4 an integer solution exists if and only if there exists a real solution. Therefore, the existence of an integer solution can be decided in polynomial time.

(ii) Let $WX \geq 0$ be a linear inequality in $\mathbb{W}$. By Lemma 6.2.3 and the definition of the degenerate system $\mathbb{W}^=$, this inequality belongs to $\mathbb{W}^=$ if and only if $\mathbb{W} \cup \{WX > 0\}$ has no solution. By (i) this can be checked in polynomial time. □

## 6.3 States

In Section 6.5 we will present an algorithm for orientability by Knuth-Bendix orders. This algorithm will work on *states* which generalize systems of rewrite rules in several ways. A state will use a generalization of rewrite rules to tuples of terms and some information about possible solutions.

Let $\succ$ be any order on ground terms. We extend it lexicographically to an order on tuples of ground terms as follows: we write $\langle l_1, \ldots, l_n \rangle \succ \langle r_1, \ldots, r_n \rangle$ if for some $i \in \{1, \ldots, n\}$ we have $l_1 = r_1, \ldots, l_{i-1} = r_{i-1}$ and $l_i \succ r_i$. We call a *tuple inequality* any expression $\langle l_1, \ldots, l_n \rangle > \langle r_1, \ldots, r_n \rangle$. The *length* of this tuple inequality is $n$.

In the sequel we assume that $\Sigma$ is a fixed signature and $e$ is a constant not belonging to $\Sigma$. The constant $e$ will play the role of a temporary substitute for a constant of the minimal weight. We also assume that different rewrite rules have disjoint sets of variables. This can be achieved by renaming variables.

We will present the algorithm for orienting a system of rewrite rules as a sequence of state changes. We call a *state* a tuple $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$, where

1. $\mathbb{R}$ is a set of tuple inequalities $\langle l_1, \ldots, l_n \rangle > \langle r_1, \ldots, r_n \rangle$, such that every two different tuple inequalities in this set have disjoint variables.

2. $\mathbb{M}$ is a set of variables. This set denotes the variables ranging over the terms of the minimal weight.

3. $\mathbb{W}$ is a system of homogeneous linear inequalities over the following variables: $\{w_g \mid g \in \Sigma \cup \{e\}\}$. This system denotes constraints on the weight function collected so far, and $w_e$ denotes the minimal weight of terms.

4. $\mathbb{U}$ is one of the following values *one* or *any*. The value *one* signals that there exists exactly one term of the minimal weight, while *any* means that no constraints on the number of elements of the minimal weight have been imposed.

5. $\mathbb{G}$ and $\mathbb{L}$ are sets of constants, each of them contains at most one element. If $d \in \mathbb{G}$ (respectively $d \in \mathbb{L}$), this signals that $d$ is the greatest (respectively least) term among the terms of the minimal weight.

6. $\ggg$ is a binary relation on $\Sigma$. This relation denotes the subset of the precedence relation computed so far.

Let $w$ be a weight function on $\Sigma$, $\gg$ a precedence relation on $\Sigma$ compatible with $w$, and $\succ$ the Knuth-Bendix order induced by $(w, \gg)$. A substitution $\sigma$ grounding for a set of variables $X$ is said to be *minimal for* $X$ if for every variable $x \in X$ the term $\sigma(x)$ is of the minimal weight. We extend $w$ to $e$ by defining $w(e)$ to be the minimal weight of a constant of $\Sigma$.

We say that the pair $(w, \gg)$ is a *solution* to a state $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$ if

1. For every tuple inequality $\langle l_1, \dots, l_n \rangle > \langle r_1, \dots, r_n \rangle$ in $\mathbb{R}$ and every substitution $\sigma$ grounding for this tuple inequality and minimal for $\mathbb{M}$ we have $\langle l_1\sigma, \dots, l_n\sigma \rangle \succ \langle r_1\sigma, \dots, r_n\sigma \rangle$.

2. The weight function $w$ solves every inequality in $\mathbb{W}$ in the following sense: replacement of each $w_g$ by $w(g)$ gives a tautology. In addition, $w(e)$ coincides with the minimal weight $w(c)$ of constants $c \in \Sigma$.

3. If $\mathbb{U} = \mathit{one}$, then there exists exactly one term of the minimal weight.

4. If $d \in \mathbb{G}$ (respectively $d \in \mathbb{L}$) for some constant $d$, then $d$ is the greatest (respectively least) term among the terms of the minimal weight. Note that if $d$ is the greatest term of the minimal weight, then the signature contains no unary function symbol of the weight 0.

5. $\gg$ extends $\ggg$.

We will now show how to reduce the orientability problem for the systems of rewrite rules to the solvability problem for states.

Let $R$ be a system of rewrite rules such that every two different rules in $R$ have disjoint variables. Denote by $\mathbb{S}_R$ the state $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$ defined as follows.

1. $\mathbb{R}$ consists of all tuple inequalities $\langle l \rangle > \langle r \rangle$ such that $l \to r$ belongs to $R$.

2. $\mathbb{M} = \emptyset$.

3. $\mathbb{W}$ consists of (a) all inequalities $w_g \geq 0$, where $g \in \Sigma$ is a non-constant; (b) the inequality $w_e > 0$ and all inequalities $w_d - w_e \geq 0$, where $d$ is a constant of $\Sigma$.

4. $\mathbb{U} =$ *any*.

5. $\mathbb{G} = \mathbb{L} = \emptyset$.

6. $\ggg$ is the empty binary relation on $\Sigma$.

LEMMA 6.3.1 *Let $w$ be a weight function, $\gg$ a precedence relation on $\Sigma$ compatible with $w$, and $\succ$ a Knuth-Bendix order induced by $(w, \gg)$. Then $\succ$ orients $R$ if and only if $(w, \gg)$ is a solution to $\mathbb{S}_R$.* $\square$

The proof is straightforward.

## 6.4 Trivial signatures

For technical reasons, we will distinguish two kinds of signatures. Essentially, our algorithm depends on whether the weights of terms are restricted or not. For the so-called *non-trivial* signatures, the weights are not restricted. When we present the orientability algorithm for the non-trivial signatures, we will use the fact that terms of sufficiently large weights always exist. For the trivial signatures we will present a simpler orientability algorithm in Section 6.6.

A signature is called *trivial* if it contains no function symbols of arity $\geq 2$, and at most one unary function symbol. Note that a signature is non-trivial if and only if it contains either a function symbol of arity $\geq 2$ or at least two function symbols of arity 1.

LEMMA 6.4.1 *Let $\Sigma$ be a non-trivial signature and $w$ be a weight function for $\Sigma$. Then for every integer $m$ there exists a ground term of the signature $\Sigma$ such that $|t| > m$.*

PROOF. It is enough to show how for every term $t$ build a term of the weight greater than $|t|$. Note that the weight of any term is positive. If $\Sigma$ contains a function symbol $g$ of arity $n \geq 2$, then $|g(t, \ldots, t)| = w(g) + n \cdot |t| > |t|$. If $\Sigma$ contains two unary function symbols, then for at least one of them $g$ we have $w(g) > 0$. Then $|g(t)| = w(g) + |t| > |t|$. $\square$

## 6.5 An algorithm for orientability in the case of non-trivial signatures

In this section we only consider non-trivial signatures. An algorithm for trivial signatures is given in Section 6.6. The algorithm given in this section will be illustrated below in Section 6.5.5 on the rewrite rule of Example 6.1.3.

Our algorithm works as follows. Given a system $R$ of rewrite rules, we build the initial state $\mathbb{S}_R = (\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$. Then we repeatedly transform $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$ as described below. We call the *size* of the state the total number of occurrences of function symbols and variables in $\mathbb{R}$. Every transformation step will terminate with either success or failure, or else decrease the size of $\mathbb{R}$.

At each step we assume that $\mathbb{R}$ consists of $k$ tuple inequalities

$$\begin{gathered}\langle l_1, L_1\rangle > \langle r_1, R_1\rangle,\\ \cdots \\ \langle l_k, L_k\rangle > \langle r_k, R_k\rangle,\end{gathered} \tag{6.3}$$

such that all of the $L_i, R_i$ are tuples of terms.

We will label parts of the algorithm. These labels will be used in the proof of its soundness. The algorithm can make a non-deterministic choice of a constant of the minimal weight, but at most once at step (T3) below, and the number of non-deterministic branches is bounded by the number of constants in $\Sigma$. If we allow to extend our signature with an extra constant, which is appropriate for most applications, then this non-deterministic choice can be replaced by by adding $e$ as a new constant in our signature.

When the set $\mathbb{W}$ of linear inequalities changes, we assume that we check the new set for satisfiability, and terminate with failure if it is unsatisfiable. Likewise, when we change $\ggg$, we check if it can be extended to an order and terminate with failure if it cannot.

### 6.5.1 The algorithm

The algorithm works as follows. Every step consists of a number of state transformations, beginning with PREPROCESS defined below. During the algorithm, we will perform two kinds of *consistency checks*:

- The *consistency check on* $\mathbb{W}$ is the check whether $\mathbb{W}$ has a solution. If it does not, we terminate with failure.
- The *consistency check on* $\ggg$ is the check whether $\ggg$ can be extended to an order, i.e., the transitive closure $\gg$ of $\ggg$ is irreflexive, i.e., for no $g \in \Sigma$ we have $g \gg g$. If $\ggg$ cannot be extended to an order, we terminate with failure.

It is not hard to argue that both kinds of consistency checks can be performed in polynomial time. The consistency check on $\mathbb{W}$ is polynomial by Lemma 6.2.7. The consistency check on $\ggg$ is polynomial since the transitive closure of a binary relation can be computed in polynomial time, see, e.g., [Cormen, Leiserson and Rivest 1991].

**PREPROCESS.** Do the following transformations while possible. If $\mathbb{R}$ contains a tuple inequality $\langle l_1, \ldots, l_n \rangle > \langle l_1, \ldots, l_n \rangle$, terminate with failure. Otherwise, if $\mathbb{R}$ contains a tuple inequality $\langle l, l_1, \ldots, l_n \rangle > \langle l, r_1, \ldots, r_n \rangle$, replace it by $\langle l_1, \ldots, l_n \rangle > \langle r_1, \ldots, r_n \rangle$.

If $\mathbb{R}$ becomes empty, proceed to TERMINATE, otherwise continue with MAIN.

**MAIN.** Now we can assume that in (6.3) each $l_i$ is a term different from the corresponding term $r_i$. For every variable $x$ and term $t$ denote by $n(x, t)$ the number of occurrences of $x$ in $t$. For example, $n(x, g(x, h(y, x))) = 2$. Likewise, for every function symbol $g \in \Sigma$ and term $t$ denote by $n(g, t)$ the number of occurrences of $g$ in $t$. For example, $n(h, g(x, h(y, x))) = 1$.

**(M1)** For all $x$ and $i$ such that $n(x, l_i) > n(x, r_i)$, add $x$ to $\mathbb{M}$.

**(M2)** If for some $i$ there exists a variable $x \notin \mathbb{M}$ such that $n(x, l_i) < n(x, r_i)$, then terminate with failure.

For every pair of terms $l, r$, denote by $W(l, r)$ the linear inequality obtained as follows. Let $v_l$ and $v_r$ be the numbers of occurrences of variables in $l$ and $r$ respectively. Then

$$W(l, r) = \sum_{g \in \Sigma} (n(g, l) - n(g, r)) w_g + (v_l - v_r) w_e \geq 0. \tag{6.4}$$

For example, if $l = h(x, f(y))$ and $r = f(g(x, g(x, y)))$, then

$$W(l, r) = w_h - 2 \cdot w_g - w_e \geq 0.$$

**(M3)** Add to $\mathbb{W}$ all the linear inequalities $W(l_i, r_i)$ for all $i$ and perform the consistency check on $\mathbb{W}$.

Now compute $\mathbb{W}^=$. If $\mathbb{W}^=$ contains none of the inequalities $W(l_i, r_i)$, proceed to TERMINATE. Otherwise, for all $i$ such that $W(l_i, r_i) \in \mathbb{W}^=$ apply the applicable case below, depending on the form of $l_i$ and $r_i$.

**(M4)** If $(l_i, r_i)$ has the form $(g(s_1, \ldots, s_n), h(t_1, \ldots, t_p))$, where $g$ is different from $h$, then extend $\ggg$ by adding $g \ggg h$ and remove the tuple inequality $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ from $\mathbb{R}$. Perform the consistency check on $\ggg$.

**(M5)** If $(l_i, r_i)$ has the form $(g(s_1, \ldots, s_n), g(t_1, \ldots, t_n))$, then replace $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ by $\langle s_1, \ldots, s_n, L_i \rangle > \langle t_1, \ldots, t_n, R_i \rangle$.

**(M6)** If $(l_i, r_i)$ has the form $(x, y)$, where $x$ and $y$ are different variables, do the following. (Note that at this point $x, y \in \mathbb{M}$.) If $L_i$ is empty, then terminate with failure. Otherwise, set $\mathbb{U}$ to *one* and replace $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ by $\langle L_i \rangle > \langle R_i \rangle$.

**(M7)** If $(l_i, r_i)$ has the form $(x, t)$, where $t$ is not a variable, do the following. If $t$ is not a constant, or $L_i$ is empty, then terminate with failure. So assume that $t$ is a constant $c$. If $\mathbb{L} = \{d\}$ for some $d$ different from $c$, then terminate with failure. Otherwise, set $\mathbb{L}$ to $\{c\}$. Replace in $L_i$ and $R_i$ the variable $x$ by $c$, obtaining $L_i'$ and $R_i'$ respectively, and then replace $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ by $\langle L_i' \rangle > \langle R_i' \rangle$.

**(M8)** If $(l_i, r_i)$ has the form $(t, x)$, where $t$ is not a variable, do the following. If $t$ contains $x$, remove $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ from $\mathbb{R}$. Otherwise, if $t$ is a non-constant or $L_i$ is empty, terminate with failure. (Note that at this point $x \in \mathbb{M}$ and $W(t, x) \in \mathbb{W}^=$.) Let now $t$ be a constant $c$. If $\mathbb{G} = \{d\}$ for some $d$ different from $c$, then terminate with failure. Otherwise, set $\mathbb{G}$ to $\{c\}$. Replace in $L_i$ and $R_i$ the variable $x$ by $c$, obtaining $L_i'$ and $R_i'$ respectively, and then replace $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ by $\langle L_i' \rangle > \langle R_i' \rangle$.

After this step repeat PREPROCESS.

**TERMINATE.** Let $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$ be the current state. Do the following.

**(T1)** If $d \in \mathbb{G}$, then for all constants $c$ different from $d$ such that $w_c - w_e \geq 0$ belongs to $\mathbb{W}^=$ extend $\ggg$ by adding $d \ggg c$. Likewise, if $c \in \mathbb{L}$, then for all constants $d$ different from $c$ such that $w_d - w_e \geq 0 \in \mathbb{W}^=$ extend $\ggg$ by adding $d \ggg c$. Perform the consistency check on $\ggg$.

**(T2)** For all $f$ in $\Sigma$ do the following. If $f$ is a unary function symbol and $w_f \geq 0$ belongs to $\mathbb{W}^=$, then extend $\ggg$ by adding $f \ggg h$ for all $h \in \Sigma - \{f\}$. Perform the consistency check on $\ggg$. If $\mathbb{U} = one$ or $\mathbb{G} \neq \emptyset$, then terminate with failure.

**(T3)** If there exists no constant $c$ such that $w_c - w_e \geq 0$ is in $\mathbb{W}^=$, then non-deterministically choose a constant $c \in \Sigma$, add $w_e - w_c \geq 0$ to $\mathbb{W}$, perform the consistency check on $\mathbb{W}$ and repeat PREPROCESS.

**(T4)** If $\mathbb{U} = one$, then terminate with failure if there exists more than one constant $c$ such that $w_c - w_e \geq 0$ belongs to $\mathbb{W}^=$.

**(T5)** Terminate with success.

We will show how to build a solution at step (T5) below in Lemma 6.5.19.

### 6.5.2 Correctness

In this section we prove correctness of the algorithm. In Section 6.5.3 we show how to find a solution when the algorithm terminates with success. The correctness will follow from a series of lemmas asserting that the transformation steps performed by the algorithm preserve the set of solutions. We will use notation and terminology of the algorithm. We say that a step of the algorithm is *correct* if the set of solutions to the state before this step coincides with the set of solutions after the step. When we prove correctness of a particular step, we will always denote by $\mathbb{S} = (\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$ the state before this step, and by $\mathbb{S}'$ the state after this step. When we use substitutions in the proof, we always assume that the substitutions are grounding for the relevant terms.

The following two lemmas can be proved by a straightforward application of the definition of solution to a state.

LEMMA 6.5.1 (consistency check) *If consistency check on* $\mathbb{W}$ *or on* $\ggg$ *terminates with failure, then* $\mathbb{S}$ *has no solution.* □

LEMMA 6.5.2 *Step* PREPROCESS *is correct.* □

Let us now analyze MAIN. For every weight function $w$ and precedence relation $\gg$ compatible with $w$ we call a *counterexample* to $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ w.r.t. $(w, \gg)$ any substitution $\sigma$ minimal for $\mathbb{M}$ such that $\langle r_i\sigma, R_i\sigma\rangle \succeq \langle l_i\sigma, L_i\sigma\rangle$ for the order $\succ$ induced by $(w, \gg)$.

Denote by $\mathbb{S}^{-i}$ the state obtained from $\mathbb{S}$ by removal of the $i$th tuple inequality $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ from $\mathbb{R}$. The following lemma follows immediately from the definition of solution.

LEMMA 6.5.3 (counterexample) *If for every solution* $(w, \gg)$ *to* $\mathbb{S}^{-i}$ *there exists a counterexample to* $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ *w.r.t.* $(w, \gg)$*, then* $\mathbb{S}$ *has no solution. If for every solution* $(w, \gg)$ *to* $\mathbb{S}^{-i}$ *there exists no counterexample to the tuple inequality* $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$*, then removing this tuple inequality from* $\mathbb{R}$ *does not change the set of solutions to* $\mathbb{S}$. □

This lemma means that we can change $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ into a different tuple inequality or change $\mathbb{M}$, if we can prove that this change does not influence the existence of a counterexample.

Let $\sigma$ be a substitution, $x$ a variable and $t$ a term. Denote by $\sigma_x^t$ the substitution defined by

$$\sigma_x^t(y) = \begin{cases} \sigma(y), & \text{if } y \neq x, \\ t, & \text{if } y = x. \end{cases}$$

LEMMA 6.5.4 *Let* $w$ *be a weight function on* $\Sigma$ *and* $\gg$ *a precedence relation on* $\Sigma$ *compatible with* $w$*. Suppose also that for some* $x$ *and* $i$ *we have* $n(x, l_i) > n(x, r_i)$ *and there exists a counterexample* $\sigma$ *to* $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ *w.r.t.* $(w, \gg)$*. Then there exists a counterexample* $\sigma'$ *to* $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ *w.r.t.* $(w, \gg)$ *minimal for* $\{x\}$.

PROOF. Suppose that $\sigma$ is not minimal for $\{x\}$. Denote by $c$ a minimal constant w.r.t. $w$ and by $t$ the term $x\sigma$. Since $\sigma$ is not minimal for $\{x\}$, we have $|t| > |c|$. Consider the substitution $\sigma_x^c$. Since $\sigma$ is a counterexample, we have $|r_i\sigma| \geq |l_i\sigma|$. We have

$$
\begin{array}{rcl}
|l_i\sigma_x^c| & = & |l_i\sigma| - n(x,l_i)\cdot(|t|-|c|);\\
|r_i\sigma_x^c| & = & |r_i\sigma| - n(x,r_i)\cdot(|t|-|c|).
\end{array}
$$

Then

$$
\begin{array}{rclcl}
|r_i\sigma_x^c| & = & |r_i\sigma| - n(x,r_i)\cdot(|t|-|c|) & \geq & |l_i\sigma| - n(x,r_i)\cdot(|t|-|c|)\\
& > & |l_i\sigma| - n(x,l_i)\cdot(|t|-|c|) & = & |l_i\sigma_x^c|.
\end{array}
$$

Therefore, $|r_i\sigma_x^c| > |l_i\sigma_x^c|$, and so $\sigma_x^c$ is a counterexample too. □

One can immediately see that this lemma implies correctness of step (M1).

LEMMA 6.5.5 *Step (M1) is correct.*

PROOF. Evidently, every solution to $\mathbb{S}$ is also a solution to $\mathbb{S}'$. But by Lemma 6.5.4, every counterexample to $\mathbb{S}$ can be turned into a counterexample to $\mathbb{S}'$, so every solution to $\mathbb{S}'$ is also a solution to $\mathbb{S}$. □

Let us now turn to step (M2).

LEMMA 6.5.6 (M2) *If for some $i$ and $x \notin \mathbb{M}$ we have $n(x,l_i) < n(x,r_i)$, then $\mathbb{S}$ has no solution. Therefore, step (M2) is correct.*

PROOF. We show that for every $(w,\gg)$ there exists a counterexample to $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$ w.r.t. $(w,\gg)$. Let $\sigma$ be any substitution grounding for this tuple inequality. Take any term $t$ and consider the substitution $\sigma_x^t$. We have

$$|r_i\sigma_x^t| - |l_i\sigma_x^t| = |r_i\sigma| - |l_i\sigma| + (n(x,r_i) - n(x,l_i))\cdot(|t| - |x\sigma|).$$

By Lemma 6.4.1 there exist terms of an arbitrarily large weight, so for a term $t$ of a large enough weight we have $|r_i\sigma_x^t| > |l_i\sigma_x^t|$, and so $\sigma_x^t$ is a counterexample to $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$.

Correctness of (M2) is straightforward. □

Note that after step (M2) for all $i$ and $x \notin \mathbb{M}$ we have $n(x,l_i) = n(x,r_i)$.

Denote by $\Theta_c$ the substitution such that $\Theta_c(x) = c$ for every variable $x$.

LEMMA 6.5.7 (M3) *Let for all $i$ and $x \notin \mathbb{M}$ we have $n(x,l_i) = n(x,r_i)$. Every solution $(w,\gg)$ to $\mathbb{S}$ is also a solution to $W(l_i,r_i)$. Therefore, step (M3) is correct.*

PROOF. Let $c$ be a constant of the minimal weight. Consider the substitution $\Theta_c$. Note that this substitution is minimal for $\mathbb{M}$. It follows from the definition of $W$ that $(w, \gg)$ is a solution to $W(l_i, r_i)$ if and only if $|l_i\Theta_c| \geq |r_i\Theta_c|$. But $|l_i\Theta_c| \geq |r_i\Theta_c|$ is a straightforward consequence of the definition of solutions to tuple inequalities.

Correctness of (M3) is straightforward. □

LEMMA 6.5.8 *Let for all $x \notin \mathbb{M}$ we have $n(x, l_i) = n(x, r_i)$. Let also $W(l_i, r_i) \in \mathbb{W}^=$. Then for every solution to $\mathbb{S}^{-i}$ and every substitution $\sigma$ minimal for $\mathbb{M}$ we have $|l_i\sigma| = |r_i\sigma|$.*

PROOF. Using the fact that $n(x, l_i) = n(x, r_i)$ for all $x \notin \mathbb{M}$, it is not hard to argue that $|l_i\sigma| - |r_i\sigma|$ does not depend on $\sigma$, whenever $\sigma$ is minimal for $\mathbb{M}$.

Let $c$ be a constant of the minimal weight. It follows from the definition of $W$ that if $W(l_i, r_i) \in \mathbb{W}^=$, then for every solution to $\mathbb{W}$ (and so for every solution to $\mathbb{S}^{-i}$) we have $|l_i\Theta_c| = |r_i\Theta_c|$. Therefore, $|l_i\sigma| = |r_i\sigma|$ for all substitutions $\sigma$ minimal for $\mathbb{M}$. □

The proof of correctness of steps (M4)–(M8) will use this lemma in the following way. A pair $(w, \gg)$ is a solution to $\mathbb{S}$ if and only if it is a solution to $\mathbb{S}^{-i}$ and a solution to $\langle l_i, L_i\rangle > \langle r_i, R_i\rangle$. Equivalently, $(w, \gg)$ is a solution to $\mathbb{S}$ if and only if it is a solution to $\mathbb{S}^{-i}$ and for every substitution $\sigma$ minimal for $\mathbb{M}$ we have $\langle l_i\sigma, L_i\sigma\rangle \succ \langle r_i\sigma, R_i\sigma\rangle$. But by Lemma 6.5.8 we have $|l_i\sigma| = |r_i\sigma|$, so $\langle l_i\sigma, L_i\sigma\rangle \succ \langle r_i\sigma, R_i\sigma\rangle$ must be satisfied by either condition 2 or condition 3 of the definition of Knuth-Bendix orders (Definition 3.3.8).

This consideration can be summarized as follows.

LEMMA 6.5.9 *Let for all $x \notin \mathbb{M}$ we have $n(x, l_i) = n(x, r_i)$. Let also $W(l_i, r_i) \in \mathbb{W}^=$. Then a pair $(w, \gg)$ is a solution to $\mathbb{S}$ if and only if it is a solution to $\mathbb{S}^{-i}$ and for every substitution $\sigma$ minimal for $\mathbb{M}$ the following holds. Let $l_i\sigma = g(t_1, \ldots, t_n)$ and $r_i\sigma = h(s_1, \ldots, s_p)$. Then at least one of the following conditions holds*

1. *$l_i\sigma = r_i\sigma$ and $L_i\sigma \succ R_i\sigma$; or*
2. *$g \gg h$; or*
3. *$g = h$ and for some $1 \leq i \leq n$ we have $t_1\sigma = s_1\sigma, \ldots, t_{i-1}\sigma = s_{i-1}\sigma$ and $t_i\sigma \succ_{KBO} s_i\sigma$.*

□

LEMMA 6.5.10 *Step (M4) is correct.*

PROOF. We know that $l_i = g(s_1, \ldots, s_n)$ and $r_i = h(t_1, \ldots, t_p)$ for $g \neq h$. Take any substitution $\sigma$ minimal for $\mathbb{M}$. Obviously, $l_i\sigma = r_i\sigma$ is impossible, so $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$ if and only if $l_i\sigma \succ r_i\sigma$. By Lemma 6.5.9 this holds if and only if $g \gg h$, so step (M4) is correct. □

LEMMA 6.5.11 *Step (M5) is correct.*

PROOF. We know that $l_i = g(s_1, \ldots, s_n)$ and $r_i = g(t_1, \ldots, t_n)$. Note that due to PREPROCESS, $l_i \neq r_i$, so $n \geq 1$. It follows from Lemma 6.5.9 that $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$ if and only if $\langle s_1, \ldots, s_n, L_i\rangle\sigma \succ \langle t_1, \ldots, t_n, R_i\rangle\sigma$, so step (M5) is correct. □

LEMMA 6.5.12 *Step (M6) is correct.*

PROOF. We know that $l_i = x$ and $r_i = y$, where $x, y$ are different variables. Note that if $L_i$ is empty, then the substitution $\Theta_c$, where $c$ is of the minimal weight, is a counterexample to $\langle x, L_i\rangle > \langle y, R_i\rangle$. So assume that $L_i$ is non-empty and consider two cases.

1. If there exist at least two terms $s, t$ of the minimal weight, then there exists a counterexample to $\langle x, L_i\rangle > \langle y, R_i\rangle$. Indeed, if $s \succ t$, then $y\sigma \succ x\sigma$ for every $\sigma$ such that $\sigma(x) = t$ and $\sigma(y) = s$.

2. If there exists exactly one term $t$ of the minimal weight, then $x\sigma = y\sigma$ for every $\sigma$ minimal for $\mathbb{M}$. Therefore, $\langle x, L_i\rangle > \langle y, R_i\rangle$ is equivalent to $\langle L_i\rangle > \langle R_i\rangle$.

In either case it is not hard to argue that step (M6) is correct. □

LEMMA 6.5.13 *Step (M7) is correct.*

PROOF. We know that $l_i = x$ and $r_i = t$. Let $c$ be the least constant in the signature. If $t \neq c$, then $\Theta_c$ is obviously a counterexample to $\langle x, L_i\rangle > \langle t, R_i\rangle$. Otherwise $t = c$, then for every counterexample $\sigma$ we have $\sigma(x) = c$. In either case it is not hard to argue that step (M7) is correct. □

LEMMA 6.5.14 *Step (M8) is correct.*

PROOF. We know that $l_i = t$ and $r_i = x$. Note that $t \neq x$ due to the PREPROCESS step, so if $x$ occurs in $t$ we have $t\sigma \succ x\sigma$ for all $\sigma$. Assume now that $x$ does not occur in $t$. Then $x \in \mathbb{M}$. Consider two cases.

1. $t$ is a non-constant. For every substitution $\sigma$ minimal for $\mathbb{M}$ we have $|t\sigma| = |x\sigma|$, hence $t\sigma$ is a non-constant term of the minimal weight. This implies that the signature contains a unary function symbol $f$ of the weight 0. Take any substitution $\sigma$. It is not hard to argue that $\sigma_x^{f(t)\sigma}$ is a counterexample to $\langle t, L_i \rangle > \langle x, R_i \rangle$.

2. $t$ is a constant $c$. Let $d$ be the greatest constant in the signature among the constants of the minimal weight. If $d \neq c$, then $\Theta_d$ is obviously a counterexample to $\langle c, L_i \rangle > \langle x, R_i \rangle$. Otherwise $d = c$, then for every counterexample $\sigma$ we have $\sigma(x) = c$.

In either case it is not hard to argue that step (M8) is correct. □

Let us now analyze steps TERMINATE. Note that for every constant $c$ the inequality $w_c - w_e \geq 0$ belongs to $\mathbb{W}$ and for every function symbol $g$ the inequality $w_g \geq 0$ belongs to $\mathbb{W}$ too.

LEMMA 6.5.15 *Step (T1) is correct.*

PROOF. Suppose $d \in \mathbb{G}$, $c \neq d$, and $w_c - w_e \geq 0$ belongs to $\mathbb{W}^=$. Then for every solution to $\mathbb{S}$ we have $w(c) = w(e)$, and therefore $c$ is a constant of the minimal weight. But since for every solution $d$ is the greatest constant among those having the minimal weight, we must have $d \gg c$.

The case $c \in \mathbb{L}$ is similar. □

LEMMA 6.5.16 *Step (T2) is correct.*

PROOF. If $f$ is a unary function symbol and $w_f \geq 0$ belongs to $\mathbb{W}^=$, then for every solution $w(f) = 0$. By the definition of Knuth-Bendix orders we must have $f \gg g$ for all $g \in \Sigma - \{f\}$. But then (i) there exists an infinite number of terms of the minimal weight and (ii) a constant $d \in \mathbb{G}$ cannot be the greatest term of the minimal weight (since for example $f(d) \succ d$ and $|f(d)| = |d|$). □

Step (T3) makes a non-deterministic choice, which can result in several states $\mathbb{S}_1, \ldots, \mathbb{S}_n$. We say that such a step is correct if the set of solutions to $\mathbb{S}$ is the union of the sets of solutions to $\mathbb{S}_1, \ldots, \mathbb{S}_n$.

LEMMA 6.5.17 *Step (T3) is correct.*

PROOF. Note that $w$ is a solution to $w_e - w_c \geq 0$ if and only if $w(c)$ is the minimal weight, so addition of $w_e - w_c \geq 0$ to $\mathbb{W}$ amounts to stating that $c$ has the minimal weight. Evidently, for every solution, there must be a constant $c$ of the minimal weight, so the step is correct. $\square$

LEMMA 6.5.18 *Step (T4) is correct.*

PROOF. Suppose $\mathbb{U} = one$, then for every solution there exists a unique term of the minimal weight. If, $c$ is a constant such that $w_c - w_e \geq 0$ belongs to $\mathbb{W}^=$, then $c$ must be a term of the minimal weight. Therefore, there cannot be more than one such a constant $c$. $\square$

### 6.5.3 Extracting a solution

In this section we will show how to find a solution when the algorithm terminates with success.

LEMMA 6.5.19 *Step (T5) is correct.*

PROOF. To prove correctness of (T5) we have to show the existence of solution. In fact, we will show how to build a particular solution.

Note that when we terminate at step (T5), the system $\mathbb{W}$ is solvable, since it was solvable initially and we performed consistency checks on every change of $\mathbb{W}$.

By Lemma 6.2.5 there exists an integer solution $w$ to $\mathbb{W}$ which is also a solution to the strict versions of every inequality in $\mathbb{W} - \mathbb{W}^=$. Likewise, there exists a linear order $\gg$ extending $\ggg$, since we performed consistency checks on every change of $\ggg$. We claim that $(w, \gg)$ is a solution to $(\mathbb{R}, \mathbb{M}, \mathbb{W}, \mathbb{U}, \mathbb{G}, \mathbb{L}, \ggg)$. To this end we have to show that $w$ is weight function, $\gg$ is compatible with $w$ and all items 1–5 of the definition of solution are satisfied.

Let us first show that $w$ is a weight function. Note that $\mathbb{W}$ contains all inequalities $w_g \geq 0$, where $g \in \Sigma$ is a non-constant, the inequality $w_e > 0$ and the inequalities $w_d - w_e \geq 0$ for every constant $d \in \Sigma$. So to show that $w$ is a weight function it remains to show that at most one unary function symbol $f$ has weight 0. Indeed, if there were two such function symbols $f_1$ and $f_2$, then at step (T2) we would add both $f_1 \ggg f_2$ and $f_2 \ggg f_1$, but the following consistency check on $\ggg$ would fail.

The proof that $\gg$ is compatible with $w$ is similar.

Denote by $\succ$ the Knuth-Bendix order induced by $(w, \gg)$.

1. *For every tuple inequality $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ in $\mathbb{R}$ and every substitution $\sigma$ minimal for $\mathbb{M}$ we have $\langle l_i\sigma, L_i\sigma \rangle \succ \langle r_i\sigma, R_i\sigma \rangle$.* In the proof we will use the fact that $w(e)$ is the minimal weight.

   By step (M3), the inequality $W(l_i, r_i)$ does not belong to $\mathbb{W}^=$ (otherwise $\langle l_i, L_i \rangle > \langle r_i, R_i \rangle$ would be removed at one of steps (M4)–(M8)). It follows from the definition of $W$ and the construction of $w$ that if $W(l_i, r_i) \in \mathbb{W} - \mathbb{W}^=$, then $|l_i\Theta_c| > |r_i\Theta_c|$, where $c$ is any constant of the minimal weight. In Lemma 6.5.8 we proved that $|l_i\sigma| - |r_i\sigma|$ does not depend on $\sigma$, whenever $\sigma$ is minimal for $\mathbb{M}$. Therefore, $|l_i\sigma| > |r_i\sigma|$ for all substitutions $\sigma$ minimal for $\mathbb{M}$.

2. *The weight function $w$ solves every inequality in $\mathbb{W}$ and $w(e)$ coincides with the minimal weight.* This follows immediately from our construction, if we show that $w(e)$ is the minimal weight. Let us show that $w_e$ is the minimal weight. Indeed, since $\mathbb{W}$ initially contains the inequalities $w_c - w_e \geq 0$ for all constants $c$, we have that $w(e)$ is less than or equal to the minimal weight. By step (T3), there exists a constant $c$ such that $w_c - w_e \geq 0$ is in $\mathbb{W}^=$, hence $w(c) = w(e)$, and so $w(e)$ is greater than or equal to the minimal weight.

3. *If $\mathbb{U} = one$, then there exists exactly one term of the minimal weight.* Assume $\mathbb{U} = one$. We have to show that (i) there exists no unary function symbol $f$ of weight 0 and (ii) there exists exactly one constant of the minimal weight. Let $f$ be a unary function symbol. By our construction, $w_f \geq 0$ belongs to $\mathbb{W}$. By step (T2) $w_f \geq 0$ does not belong to $\mathbb{W}^=$, so by the definition of $w$ we have $w(f) > 0$. By our construction, $w_c - w_e \geq 0$ belongs to $\mathbb{W}$ for every constant $c$. By step (T4), at most one of such inequalities belongs to $\mathbb{W}^=$. But if $w_c - w_e \geq 0$ does not belong to $\mathbb{W}^=$, then $w(c) - w(e) > 0$

by the construction of $w$. Therefore, there exists at most one constant of the minimal weight.

4. *If $d \in \mathbb{G}$ (respectively $d \in \mathbb{L}$) for some constant $d$, then $d$ is the greatest (respectively least) term among the terms of the minimal weight.* We consider the case $d \in \mathbb{G}$, the case $d \in \mathbb{L}$ is similar. But by step (T2) there is no unary function symbol $f$ such that $w_f \geq 0$ belongs to $\mathbb{W}^=$, therefore $w(f) > 0$ for all unary function symbols $f$. This implies that only constants may have the minimal weight. But by step (T1) and the definition of $w$, for all constants $c$ of the minimal weight we have $d \ggg c$, and hence also $d \gg c$.

5. *$\gg$ extends $\ggg$.* This follows immediately from our construction.

□

### 6.5.4 Time complexity

Provided that we use a polynomial-time algorithm for solving homogeneous linear inequalities, and a polynomial-time algorithm for transitive closure, we can prove the following lemma.

LEMMA 6.5.20 *The algorithm runs in time polynomial of the size of the system of rewrite rules.*

PROOF. Note that the algorithm makes polynomial number of steps. Indeed, initially the size of $\mathbb{R}$ is $O(n \log n)$ of the size of the system of rewrite rules (and can even be made linear, if we avoid renaming variables). Each of the steps (M4)–(M8) decreases the size of $\mathbb{R}$. The algorithm can make a non-deterministic choice, but at most once, and the number of non-deterministic branches is bounded by the number of constants, so it is linear in the size of the original system.

We proved that the number of steps is polynomial in the size of the input. It remains to prove that every step can be made in polynomial time of the size of a state and that the size of every state is polynomial in the size of the input.

Solvability of $\mathbb{W}$ can be checked in polynomial time by Lemma 6.2.7. The system $\mathbb{W}^=$ can be built in polynomial time by the same lemma. The relation $\ggg$ can be extended to an order if and only if the transitive closure $\ggg'$ of $\ggg$

is irreflexive, i.e., there is no $g$ such that $g \ggg' g$. The transitive closure can be built in polynomial time. The check for irreflexivity can be obviously done in polynomial time too. Therefore, every step can be performed in polynomial time of the size of the state.

It remains to show that the size of $\mathbb{S}$ is bound by a polynomial. The only part of $\mathbb{S}$ that is not immediately seen to be polynomial is $\mathbb{W}$. However, it is not hard to argue that the number of equations in $\mathbb{S}$ of the form $W(l, r)$ is bound by the size of the input, and every equation obviously has a polynomial size. It is also easy to see that the size of the remaining equations is polynomial too. $\square$

### 6.5.5 A simple example

Let us consider how the algorithm works on the rewrite rule $g(x, a, b) \to g(b, b, a)$ of Example 6.1.3. Initially, $\mathbb{R}$ consists of one tuple inequality

$$\langle g(x, a, b)\rangle > \langle g(b, b, a)\rangle \tag{6.5}$$

and $\mathbb{W}$ consists of the following linear inequalities:

$$w_g \geq 0, \quad w_e > 0, \quad w_a - w_e \geq 0, \quad w_b - w_e \geq 0.$$

At step (M1) we note that $n(x, g(x, a, b)) = 1 > 0 = n(x, g(b, b, a))$. Therefore, we add $x$ to $\mathbb{M}$.

At step (M3) we add the linear inequality $w_e - w_b \geq 0$ to $\mathbb{W}$ obtaining

$$w_g \geq 0, \quad w_e > 0, \quad w_a - w_e \geq 0, \quad w_b - w_e \geq 0, \quad w_e - w_b \geq 0.$$

Now we compute $\mathbb{W}^=$. It consists of two equations $w_b - w_e \geq 0$ and $w_e - w_b \geq 0$, so we have to apply one of the steps (M4)–(M8), in this case the applicable step is (M5). We replace (6.5) by

$$\langle x, a, b\rangle > \langle b, b, a\rangle. \tag{6.6}$$

At the next iteration of step (M3) we should add to $\mathbb{W}$ the linear inequality $w_e - w_b \geq 0$, but this linear inequality is already a member of $\mathbb{W}$, and moreover a member of $\mathbb{W}^=$. So we proceed to step (M7). At this step we set $\mathbb{L} = \{b\}$ and replace (6.6) by

$$\langle a, b \rangle > \langle b, a \rangle. \tag{6.7}$$

Then at step (M2) we add $w_a - w_b \geq 0$ to $\mathbb{W}$ obtaining

$$w_g \geq 0, \quad w_e > 0, \quad w_a - w_e \geq 0, \quad w_b - w_e \geq 0, \quad w_e - w_b \geq 0, \quad w_a - w_b \geq 0.$$

Now $w_a - w_b \geq 0$ does not belong to the degenerate subsystem of $\mathbb{W}$, so we proceed to TERMINATE. Steps (T1)–(T4) change neither $\mathbb{W}$ nor $\ggg$, so we terminate with success.

Solutions extracted according to Lemma 6.5.19 will be any pairs $(w, \gg)$ such that $w(a) > w(b)$. Note that these are *not all* solutions. There are also solutions such that $w(a) = w(b)$ and $a \gg b$. However, if we try to find a description of all solutions we cannot any more guarantee that the algorithm runs in polynomial time.

## 6.6 Orientability for trivial signatures

Consider a trivial signature which consists of a unary function symbol $g$ and some constants. Let $R$ be a system of rewrite rules in this signature. If some rule in $R$ has the form $t \to g^n(x)$ such that $x$ does not occur in $t$, then the system is evidently not orientable. If $R$ contains no such rule, then $R$ can be replaced by an equally orientable ground system, as the following lemma shows.

LEMMA 6.6.1 *Let $R$ be a system of rewrite rules in a trivial signature $\Sigma$ such that no rule in $R$ contains a variable occurring in its right-hand side but not the left-hand side. Define the ground system $R'$ obtained from $R$ by the following transformations:*

1. *Replace every rule $g^m(x) \to g^n(d)$ in $R$ by all rules $g^m(c) \to g^n(d)$ such that $c$ is a constant in $\Sigma$.*
2. *For every rule $g^m(x) \to g^n(x)$ in $R$, if $m > n$ then remove this rule, otherwise terminate with failure.*

*Then a Knuth-Bendix order $\succ$ orients $R$ if and only if it orients $R'$.*

□

We leave the proof of this lemma to the reader. Note that the size of $R'$ in the lemma is polynomial in the sum of the sizes of $R$ and $\Sigma$. Therefore, we can restrict ourselves to ground systems.

Moreover, we can assume that for every rule in $R'$ the function symbol $g$ never occurs in both left-hand side and right-hand side of $R$. Indeed, this can be achieved by replacing every rewrite rule $g(s) \to g(t)$ in $R'$ by $s \to t$ until $g$ occurs in at most one side of the rule. Evidently, we can assume that $R'$ contains no trivial rules $c \to c$. So we obtain a system consisting of rules $g^n(c) \to d$, $c \to g^n(d)$, where $n > 0$, or $c \to d$ such that $c, d$ are different constants. In other words, for every rule $l \to r$ in $R'$ the outermost symbol of $l$ is different from the outermost symbol of $r$.

In order to check orientability of $R'$, consider the system of homogeneous linear inequalities $\mathbb{W}$ which consists of

1. the inequalities $w_c > 0$ for all constants $c \in \Sigma$ and the inequality $w_g \geq 0$;
2. for every rule $l \to r$ in $R'$ the inequalities $W(l, r) = \sum_{h \in \Sigma}(n(h, l) - n(h, r))w_h \geq 0$.

Evidently, $\mathbb{W}$ can be built in time polynomial in the size of $R'$. Evidently, if $\mathbb{W}$ is unsatisfiable, then $R'$ is not orientable. If $\mathbb{W}$ is satisfiable, let $\mathbb{W}^=$ be the degenerate subsystem of $\mathbb{W}$. Let us build a binary relation $\ggg$ on $\Sigma$ as follows:

1. for every rule $l \to r$ in $R'$, if $W(l, r) \in \mathbb{W}^=$, then we take the outermost symbols $h_1$ and $h_2$ of $l$ and $r$ respectively and add $h_1 \ggg h_2$ to $\ggg$;
2. if $w_g \geq 0$ belongs to $\mathbb{W}^=$, then add $g \ggg c$ to $\ggg$ for all constants $c \in \Sigma$.

We leave it to the reader to check that $R'$ is orientable if and only if $\ggg$ can be extended to a linear order. We can prove in the same way as before, that the check for orientability of $R'$ can be done in polynomial time.

## 6.7 The problem of orientability by Knuth-Bendix orders is P-complete

In Section 6.5.4 we have shown that the orientability problem can be solved in polynomial time. In this section we show that this problem is P-complete, and moreover it is P-hard even for ground rewrite systems. To this end, we reduce the *circuit value problem* which is known to be P-complete (see, e.g., [Papadimitriou 1994]), to the orientability problem. Our reduction consists of two steps:

1. we reduce the problem of solving systems of linear inequalities $AX \geq \mathbf{0}$, $X > \mathbf{0}$, where $A$ is an integer matrix, to the orientability problem;

2. we reduce the circuit value problem to solvability of such systems.

In the systems of linear inequalities, we assume all coefficients to be written in the unary notation. Both reductions will be LOGSPACE.

Let $AX \geq \mathbf{0}$ be a system of linear inequalities and we are looking for strictly positive solutions to it. For every variable $x_i$ in the system we introduce a unary function symbol $f_i$. We consider the signature $\Sigma$ consisting of all such symbols $f_i$, two unary symbols $g, h$, and a constant $c$. We will construct a ground rewrite rule system $R$ whose orientability will be equivalent to the existence of a solution to $AX \geq \mathbf{0}, X > \mathbf{0}$ as follows. First of all, $R$ contains the rewrite rule

$$ghc \to hggc.$$

A Knuth-Bendix order with parameters $(w, \gg)$ orients this rule if and only if $w(g) = 0$ (and hence also $g \gg h$). For each linear inequality $I$ in the system, we add to $R$ a rewrite rule $r(I)$, which will be demonstrated by an example (in order to avoid double indices). Suppose, for example, that the inequality can be rewritten in the form

$$a_1x_1 + \ldots + a_kx_k \geq a_{k+1}x_{k+1} + \ldots + a_nx_n. \tag{6.8}$$

where $x_1, \ldots, x_n$ are different variables and $a_1, \ldots, a_n$ are non-negative coefficients. Then $r(I)$ has the form

$$ghf_1^{a_1} \cdots f_k^{a_k} c \to hgf_{k+1}^{a_{k+1}} \cdots f_n^{a_n} c \tag{6.9}$$

Note that for every solution we must have $w(f_i) > 0$ since there may be at most one function symbol of the weight 0. For every weight function $w$ consider the substitution $s$ of integers to variables such that $w(f_i) = s(x_i)$ and let $\gg$ be an arbitrary precedence relation such that $g$ is maximal w.r.t. $\gg$. We leave it to the reader to check that $(w, \gg)$ is a solution to $R$ if and only if $s$ is a solution to $AX \geq \mathbf{0}, X > \mathbf{0}$.

It is not hard to argue that the reduction of $A$ to $R$ is LOGSPACE, provided that the coefficients of the linear inequalities are written in the unary notation.

Let us now describe a reduction of the circuit value problem to the problem of whether a given system of linear integer inequalities has a positive solution. Consider a circuit with gates $g_1, \ldots, g_n$. For each gate $g_i$ we introduce a new numerical variable $x_i$. We will also use an auxiliary numerical variable $y$. We construct a system of linear integer inequalities $\mathbb{W}$ in such a way that the circuit has the value *TRUE* if and only if $\mathbb{W}$ has a positive solution. For each gate $g_i$ we introduce a system of numerical constraints $\mathbb{W}_i$ in the following way. If $g_i$ is a *FALSE* gate then $\mathbb{W}_i$ is $\{x_i = y\}$, likewise if $g_i$ is a *TRUE* gate then $\mathbb{W}_i$ is $\{x_i = 2y\}$. If $g_i$ is a *NOT* gate with an input $g_j$ then $\mathbb{W}_i$ is $\{x_i = 3y - x_j\}$. If $g_i$ is an *AND* gate with inputs $g_j$ and $g_k$ then $\mathbb{W}_i$ is $\{y \leq x_i \leq 2y,\ x_i \leq x_j,\ x_i \leq x_k,\ x_j + x_k - 2y \leq x_i\}$. Let $\mathbb{W}'$ be the union of all $\mathbb{W}_i$ for $1 \leq i \leq n$. It is straightforward to check that for every positive solution to the system $\mathbb{W}'$ each variable $x_i$ has the value of the variable $y$ or twice that value, moreover it has the value of $y$ if and only if the gate $g_i$ has the value *FALSE*. To complete the construction we obtain $\mathbb{W}$ by adding to $\mathbb{W}'$ an equation $x_n = 2y$. Note that the coefficients of $\mathbb{W}$ are small, so they can be considered as written in the unary notation.

We have shown how to reduce the circuit value problem to the orientability problem. It is clear that all reductions can be done by a logarithmic-space algorithm.

## 6.8 Solving constraints consisting of a single inequality

In Chapter 4 we show that the problem of solving Knuth-Bendix ordering constraints is NP-complete. Let us show that the problem of solving Knuth-Bendix ordering constraints consisting of a single inequality can be solved in polynomial time. Let us fix a Knuth-Bendix order on ground terms, i.e., a precedence relation on the signature $\Sigma$ and a weight function $w$. Our problem is to decide for a given pair of terms $s$ and $t$ whether there exists a grounding substitution $\sigma$ such that $s\sigma \succ_{KBO} t\sigma$. Since every Knuth-Bendix order is total on ground terms our problem is equivalent to the following problem: for a given pair of terms $t$ and $s$ decide whether for every grounding substitutions $\sigma$, $t\sigma \succeq s\sigma$ holds. The algorithm we present is similar to the algorithm for the orientability. The main difference is that there is no need to solve systems of linear inequalities for this problem. Since the order is given, we can use a simpler version of the notion of state $\mathbb{S} = (\mathbb{R}, \mathbb{M})$, where $\mathbb{R}$ is a single tuple inequality and $\mathbb{M}$ is a set of variables. Instead of tuple inequalities $\langle L \rangle > \langle R \rangle$ we will consider a new kind of tuple inequalities $\langle L \rangle \geq \langle R \rangle$ with a natural interpretation. Initially $\mathbb{R}$ consists of the tuple inequality $\langle t \rangle \geq \langle s \rangle$ and $\mathbb{M} = \emptyset$. Let $e$ denote the constant that is the minimal term w.r.t. $\succ$. Instead of using the inequality $W(l, r)$, we will use the inequality $W'(l, r) = \sum_{g \in \Sigma}(n(g, l) - n(g, r))w(g) + (v_l - v_r)w(e) \geq 0$, where $v_l$ and $v_r$ are the numbers of occurrences of variables in $l$ and $r$ respectively. Let us present the algorithm.

**PREPROCESS.** Do the following transformations while possible. If $\mathbb{R}$ has the form $\langle\rangle \geq \langle\rangle$, then terminate with success. If $\mathbb{R}$ consists of a tuple inequality $\langle l, l_1, \ldots, l_n \rangle \geq \langle l, r_1, \ldots, r_n \rangle$, replace it by $\langle l_1, \ldots, l_n \rangle \geq \langle r_1, \ldots, r_n \rangle$.

**MAIN.** Now we can assume that $\mathbb{R}$ consists of a tuple $\langle l, L \rangle \geq \langle r, R \rangle$ and the term $l$ is different from the term $r$.

**(M1)** For all $x$ such that $n(x, l) > n(x, r)$, add $x$ to $\mathbb{M}$.

**(M2)** If there exists a variable $x \notin \mathbb{M}$ such that $n(x, l) < n(x, r)$, then terminate with failure.

**(M3)** If $W'(l, r) > 0$ then terminate with success. If $W'(l, r) < 0$ then terminate with failure.

Note that at this point we have $W'(l, r) = 0$.

**(M4)** If $(l, r)$ has the form $(g(s_1, \ldots, s_n), h(t_1, \ldots, t_p))$ where $g$ and $h$ are distinct, then do the following. If $g \gg h$ terminate with success, otherwise terminate with failure.

**(M5)** If $(l, r)$ has the form $(g(s_1, \ldots, s_n), g(t_1, \ldots, t_n))$, then replace $\langle l, L \rangle \geq \langle r, R \rangle$ by $\langle s_1, \ldots, s_n, L \rangle \geq \langle t_1, \ldots, t_n, R \rangle$.

**(M6)** If $(l, r)$ has the form $(x, y)$, where $x$ and $y$ are different variables, do the following. (Note that at this point $x, y \in \mathbb{M}$.) If there exists only one term of the minimal weight, then replace $\langle l, L\rangle \geq \langle r, R\rangle$ by $\langle L\rangle \geq \langle R\rangle$. Otherwise terminate with failure.

**(M7)** If $(l, r)$ has the form $(x, t)$, where $t$ is not a variable, do the following. If $t$ is different from $e$, then terminate with failure. Otherwise, replace all occurrences of $x$ in $L$ and $R$ by $e$ obtaining $L'$ and $R'$. Replace $\langle l, L\rangle \geq \langle r, R\rangle$ by $\langle L'\rangle \geq \langle R'\rangle$.

**(M8)** If $(l, r)$ has the form $(t, x)$, where $t$ is not a variable, do the following. If $t$ contains $x$ then terminate with success. Otherwise, if $t$ is not the greatest term among the terms of the minimal weight, then terminate with failure. Otherwise, replace all occurrences of $x$ in $L$ and $R$ by $t$ obtaining $L'$ and $R'$, and replace $\langle l, L\rangle \geq \langle r, R\rangle$ by $\langle L'\rangle \geq \langle R'\rangle$. Note that this step does not increase the size of the tuple inequality since $t$ must be a constant, when we substitute it for $x$.

After this step repeat PREPROCESS.

The proof of correctness of each step is almost the same as the proof of correctness for the corresponding steps in the orientability algorithm, so we leave it to the reader. Let us estimate the complexity of this algorithm assuming a standard RAM model and considering integer addition and comparison as constant time operations. Since every iteration of the algorithm decreases the size of $\mathbb{R}$ (measured as the number of symbols), the number of iterations is at most linear in the size of the input. By the routine inspection of the steps (M1)–(M8) it is not hard to argue that every step also requires at most a linear number of elementary operations. For example, computing $n(x, l)$ and $n(x, r)$ simultaneously for all variables $x$ at the step (M1) can be done in linear time, as well as computing $W(l', r')$ at the step (M3). Therefore, our algorithm decides ordering constraints consisting of a single inequality in the time $O(n^2)$.

## 6.9 Main results

Lemmas 6.5.1–6.5.19 guarantee that the orientability algorithm is correct and Lemma 6.5.20 implies that it runs in polynomial time. Hence we obtain the following theorem.

THEOREM 6.9.1 *The problem of the existence of a Knuth-Bendix order which orients a given term rewriting system can be solved in the time polynomial in the size of the system.* □

From the reductions of Section 6.7 we also obtain the following.

THEOREM 6.9.2 *The problem of orientability of term rewriting systems by Knuth-Bendix orders is P-complete. Moreover, it is P-hard even for ground rewriting systems.* □

In Section 6.8 we proved the following theorem.

THEOREM 6.9.3 *The problem of solving a given Knuth-Bendix ordering constraint consisting of a single inequality can be solved in the time $O(n^2)$.* □

The *real-valued Knuth-Bendix order* is defined in the same way as above, except that the range of the weight function is the set of non-negative real numbers. Real-valued Knuth-Bendix orders was introduced by Martin [1987]. Note that in view of the results of Section 6.2 on systems of homogeneous linear inequalities (Lemmas 6.2.4 and 6.2.5) the algorithm is also sound and complete for the real-valued orders. Therefore, we have

THEOREM 6.9.4 *If a rewrite rule system is orientable by a real-valued Knuth-Bendix order, then it is also orientable by an integer-valued Knuth-Bendix order.* □

It follows from this theorem that all our results formulated for integer-valued Knuth-Bendix orders also hold for real-valued Knuth-Bendix orders.

It is worth noting that unlike integer-valued Knuth-Bendix orders, real-valued Knuth-Bendix orders allow one to classify and topologise the space of all simplification orders, for details see [Martin and Shand 2000].

# Chapter 7

# Orientability of equalities by Knuth-Bendix Orders

This chapter is based on the paper [Korovin and Voronkov 2003*c*].

In this chapter we extend orientability results for term rewriting systems, studied in the previous chapter, to orientability of systems consisting of equalities and term rewrite rules.

Let $\succ$ be any order on ground terms and $s \simeq t$ be an equality. We say that $\succ$ *orients an equality* $s \simeq t$, if it orients either the rewrite rule $s \to t$ or the rewrite rule $t \to s$. The *orientability problem* is a problem of determining whether there exists a simplification order which orders a given system of equalities and rewrite rules. A straightforward algorithm for checking orientability of systems of equalities would be to try all possible orientations of equalities and apply an orientability algorithm for term rewriting systems. Such an algorithm would require to test an exponential number of possible orientations of equalities. We show how to avoid this problem for Knuth-Bendix orders.

**Theorem 7.7.1** *The problem of the existence of a Knuth-Bendix order which orients a given system of equalities and rewrite rules can be solved in the time polynomial in the size of the system. Moreover, if the system of equalities and rewrite rules is orientable by a Knuth-Bendix order we can find such an order in polynomial time.*

As a basis for our orientability algorithm for systems consisting of equalities and rewrite rules we will take the orientability algorithm for systems of rewrite rules studied in Chapter 6.

We also show that orientability of systems of equalities is at least as hard as orientability of term rewriting systems.

**Theorem 7.7.2** *The problem of orientability of systems of equalities and rewrite rules by Knuth-Bendix orders is P-complete. Moreover, it is P-hard even for systems consisting only of equalities or only of rewrite rules.*

## 7.1 Preliminaries

An *equality* is a multiset of two terms $s, t$, usually denoted by $s \simeq t$. Note that $s \simeq t$ and $t \simeq s$ are regarded as the same equality. A *system of equalities and rewrite rules* is a finite set of equalities and rewrite rules. An expression $E$ (e.g. a term, equality, or a rewrite rule) is called *ground* if no variable occurs in $E$.

The following definition is central to this chapter.

Definition 7.1.1 (orientability) A Knuth-Bendix order *orients* an equality $s \simeq t$ if it orients either the rewrite rule $s \to t$ or the rewrite rule $t \to s$. A Knuth-Bendix order *orients a system* $R$ of equalities and rewrite rules if it orients every equality and rewrite rule in $R$. □

In Chapter 6 we have proved that orientability can be solved in polynomial time for systems consisting of rewrite rules only.

Let us show that the problem of orientability of systems of equalities is at least as hard as the problem of orientability of systems of rewrite rules.

Proposition 7.1.2 *There exists a logarithmic-space algorithm which for a given system of rewrite rules $R$ produces a system of equalities $E$ such that $R$ is orientable by Knuth-Bendix orders if and only if so is $E$.*

Proof. Consider a rewrite system $R$. Let $g$ be a new binary symbol and $c$ be a new constant which do not occur in $R$. Consider a rewrite system $R'$ which is obtained from $R$ by replacing each rewrite rule $l \to r$ with a rewrite rule $g(l, x) \to g(r, c)$ where $x$ is a variable which does not occur in $l \to r$. Let us check that $R$ is orientable by Knuth–Bendix orders if and only if $R'$ is. Indeed, let $\succ$ be a Knuth–Bendix order which orients $R$ then we extend parameters of this order to the new symbols in such a way that $c$ becomes a minimal term in this order. Now it is straightforward to check that the obtained order $\succ'$ orients $R'$. For the converse direction let us note that if a Knuth–Bendix order orients $R'$ then the same order also orients $R$.

To conclude the proof we consider the system of equalities $E$ induced by $R'$. Since in each rewrite rule from $R'$ there exists a variable occurring in the left hand-side and not occurring in the right hand-side it is easy to see that $E$ is orientable if and only if $R'$ is orientable. □

Note that this reduction also works for the lexicographic path orders.

## 7.2 Systems of homogeneous linear inequalities

In our proofs and in the algorithm we will use several properties of homogeneous linear inequalities. In the previous chapter, Section 6.2 we have studied properties of homogeneous linear inequalities that we will use here as well.

LEMMA 7.2.1 *Consider a system of homogeneous linear inequalities* $\mathbb{W}$ *and an integer homogeneous linear inequality* $UX > 0$. *If there exists a solution* $S$ *to the system* $\mathbb{W} \cup \{UX > 0\}$ *then the degenerate subsystem of* $\mathbb{W}$ *coincides with the degenerate subsystem of* $\mathbb{W} \cup \{UX > 0\}$.

PROOF. We can assume that $\mathbb{W}$ is of the form $AX \geq \mathbf{0}$. By Lemma 6.2.2 we can find integer vectors $V_1, \ldots, V_n$ such that the set of solutions to $AX \geq \mathbf{0}$ is (6.2). Since we have that $US > 0$ for a solution to $AX \geq \mathbf{0}$ then for some $1 \leq i \leq n$ we have $UV_i > 0$. Also from Lemma 6.2.3 we have that there exists a solution $S$ to $AX \geq \mathbf{0}$ such that for each inequality $WX \geq 0$ from the nondegenerate subsystem of $AX \geq \mathbf{0}$ we have $WS > 0$. Now we consider a positive number $r$ such that $rUV_i + US > 0$, such a number always exists since we have $UV_i > 0$. It is straightforward to check that $rV_i + S$ satisfies the required properties. □

COROLLARY 7.2.2 *Consider a system of homogeneous linear inequalities* $\mathbb{W}$, *then* $\mathbb{W}^=$ *coincides with* $(\mathbb{W}^=)^=$.

PROOF. From the previous lemma it follows that if we add to the system $\mathbb{W}^=$ an inequality from the non-degenerate subsystem of $\mathbb{W}$ then we obtain a new system with the degenerate part equal to $(\mathbb{W}^=)^=$. If we continue this process until we have added all inequalities from the non-degenerate subsystem of $\mathbb{W}$ we obtain that $\mathbb{W}^=$ coincides with $(\mathbb{W}^=)^=$. □

Let us consider a system of homogeneous linear inequalities $\mathbb{W}$. We say that an equality $VX = 0$ *follows* from $\mathbb{W}$ if for every solution $S$ to $\mathbb{W}$ we have $VS = 0$. Now our goal is to show that for every equality $VX = 0$ if it follows from $\mathbb{W}$ then it already follows from the degenerate subsystem of $\mathbb{W}$. For this we use the following theorem.

THEOREM 7.2.3 *(Fundamental theorem of linear inequalities.) Let $A_1, \ldots, A_m, U$ be vectors in $n$–dimensional space. Then, either*

1. *$U$ is a non-negative linear combination of linearly independent vectors from $A_1, \ldots, A_m$, or*
2. *there exists a vector $W$ such that $UW < 0$ and $A_i W \geq 0$ for $1 \leq i \leq m$.*

PROOF. The proof can be found in, e.g. [Schrijver 1998]. □

LEMMA 7.2.4 *Consider a system of homogeneous linear inequalities $W$ with an integer matrix and an integer homogeneous linear equality $UX = 0$. If $UX = 0$ follows from $W$ then it follows from the degenerate subsystem of $W$.*

PROOF. We can assume that $\mathbb{W}$ is of the form $AX \geq \mathbf{0}$. First we prove that if $UX = 0$ follows from $AX \geq \mathbf{0}$ then the vector $U$ is a non-negative linear combination of the row vectors of the degenerate subsystem of $AX \geq \mathbf{0}$. For this we apply Theorem 7.2.3 to the row vectors of the matrix $A$ and the vector $U$. There are two possible cases.

- $U$ is a non-negative linear combination of the row vectors from the matrix $A$. $1 \leq i \leq k$ Let us show that in this combination all coefficients of the vectors from the non-degenerate subsystem of $AX \geq \mathbf{0}$ are equal to zero. Otherwise, we consider such a vector $C$. Since $C$ is a row vector from the non-degenerate subsystem, there exists a solution $S$ to $AX \geq \mathbf{0}$ such that $CS > 0$ and therefore $US > 0$, which contradicts to the assumption that $UX = 0$ follows from $AX \geq \mathbf{0}$.

- there exists a vector $W$ such that for each row vector $Q$ of $A$ we have $QW \geq 0$ and also $UW < 0$. But this contradicts to the assumption that $UX = 0$ follows from $AX \geq \mathbf{0}$.

We have shown that $U$ is a non-negative linear combination of the row vectors from the degenerate subsystem of $AX \geq \mathbf{0}$.

Now using Corollary 7.2.2 it is easy to see that $UX = 0$ follows from the degenerate subsystem of $AX \geq \mathbf{0}$. $\Box$

## 7.3 Constraints

In Section 7.5 we will present an algorithm for orientability by Knuth-Bendix orders. The algorithm works not only with equalities and rewrite rules. It also uses linear inequalities on the weights of the signature symbols, constraints on the precedence relation, and some additional information. All this information will be formalized using the notion of *constraint*.

Let $>$ be any binary relation on ground terms. We extend it lexicographically to a relation on tuples of ground terms as follows: we have $\langle l_1, \ldots, l_n \rangle > \langle r_1, \ldots, r_n \rangle$ if for some $i \in \{1, \ldots, n\}$ we have $l_1 = r_1, \ldots, l_{i-1} = r_{i-1}$ and $l_i > r_i$.

In the sequel we assume that $\Sigma$ is a fixed signature. We also assume that different equalities and rewrite rules have disjoint sets of variables. This can be achieved by renaming variables.

Our algorithm will work on *constraints*. Orientability of a rewrite rule or an equality are special kinds of constraints. In addition, there are constraints on the precedence relation and on the weights of the symbols in $\Sigma$. The algorithm will transform constraints step by step. We will show that every step preserves satisfiability of constraints. Before defining constraints, we introduce special kind of variables, called *marked variables*. Intuitively, marked variables range only over terms of the minimal weight.

DEFINITION 7.3.1 (Constraint) An *atomic constraint* is an expression having one of the following forms:

1. $\langle l_1, \ldots, l_n \rangle \mathrel{?{\succ}} \langle r_1, \ldots, r_n \rangle$, where $l_1, \ldots, l_n, r_1, \ldots, r_n$ are terms. Such constraints are called *rewriting constraints*.

2. $\langle l_1, \ldots, l_n \rangle \mathrel{{\prec}?{\succ}} \langle r_1, \ldots, r_n \rangle$, where $l_1, \ldots, l_n, r_1, \ldots, r_n$ are terms. Such constraints are called *orientability constraints*.

3. A (strict or non-strict) homogeneous linear inequality over the variables $\{w_g \mid g \in \Sigma\}$. Such constraints are called *weight constraints*.

4. $g \mathrel{?\!\gg} h$, where $g, h \in \Sigma$. Such constraints are called *precedence constraints*.

5. $gtmw(c)$, where $c$ is a constant.

A *constraint* $C$ is a conjunction $C_1 \wedge \ldots \wedge C_n$ of (zero or more) atomic constraints. Alternatively, we will sometimes regard a constraint as the *set* $\{C_1, \ldots, C_n\}$ of all atomic constraints in it. In this case we say that $C$ *contains* the atomic constraints $C_1, \ldots, C_n$. Conjunctions (or sets) of atomic rewriting constraints are called rewriting constraints, and similar for the orientability, weight, and precedence constraints. $\square$

We consider constraints as conditions on the Knuth-Bendix order. Every Knuth-Bendix order which satisfies all atomic constraints in $C$ is called a *solution* to this constraint. In order to define solutions, let us give a technical definition. A substitution $\sigma$ is called an *admissible substitution* for a weight function $w$ if for every marked variable $x$ the term $\sigma(x)$ is a ground term of the minimal weight, that is $w(\sigma(x))$ is equal to the smallest weight of a constant in $\Sigma$.

DEFINITION 7.3.2 (Solution) Let $\succ$ be the Knuth-Bendix order induced by $(w, \gg)$. This order is called a *solution* to an atomic constraint $C$ if one of the following conditions holds.

1. $C$ is a rewriting constraint $\langle l_1, \ldots, l_n \rangle \mathrel{?\!\succ} \langle r_1, \ldots, r_n \rangle$, and for every admissible substitution $\sigma$ we have $\langle l_1\sigma, \ldots, l_n\sigma \rangle \succ \langle r_1\sigma, \ldots, r_n\sigma \rangle$.

2. $C$ is an orientability constraint $\langle l_1, \ldots, l_n \rangle \mathrel{\prec\!?\!\succ} \langle r_1, \ldots, r_n \rangle$ and $\succ$ is a solution to either $\langle l_1, \ldots, l_n \rangle \mathrel{?\!\succ} \langle r_1, \ldots, r_n \rangle$ or $\langle r_1, \ldots, r_n \rangle \mathrel{?\!\succ} \langle l_1, \ldots, l_n \rangle$.

3. $C$ is a weight constraint and $w$ solves $C$ in the following sense: replacement of each $w_g$ by $w(g)$ gives a tautology.

4. $C$ is a precedence constraint $g \mathrel{?\!\gg} h$, and $g \gg h$.

5. $C$ is a constraint $gtmw(c)$, and $c$ is the greatest term of the minimal weight.

A solution to an arbitrary constraint $C$ is a solution to every atomic constraint in $C$. A constraint $C$ is *satisfiable* if it has a solution. A constraint $C_1$ *implies* a constraint $C_2$, denoted by $C_1 \supset C_2$, if every solution to $C_1$ is also a solution to $C_2$. Two constraints are *equivalent* if they have the same solutions. $\square$

We will often write atomic constraints in $\mathbb{W}$ in an equivalent form, for example write $w_c > w_e$ instead of $w_c - w_e > 0$.

We will now show how to reduce the orientability problem for the systems of equalities and rewrite rules to the satisfiability problem for constraints.

Let $R$ be a system of equalities and rewrite rules such that every two different rules in $R$ have disjoint variables. Denote by $C_R$ the conjunction of the following constraints:

1. rewriting constraints $\langle l \rangle \mathrel{?\!\succ} \langle r \rangle$ such that $l \to r$ belongs to $R$.

2. orientability constraints $\langle l \rangle \mathrel{\prec\!?\!\succ} \langle r \rangle$ such that $l \simeq r$ belongs to $R$.

The following lemma is straightforward.

LEMMA 7.3.3 *A Knuth-Bendix order $\succ$ orients $R$ if and only if $\succ$ is a solution to $C_R$.*

□

## 7.4 Rich constraints and trivial signatures

For technical reasons, it will be convenient for us to work with constraints which contain enough information to decide some properties of its solutions, for example, which of the constants of $\Sigma$ is the smallest. Such constraints are introduced here and called *rich constraints*.

DEFINITION 7.4.1 (Rich Constraint) A constraint $C$ is called *rich* if

1. $C$ contains all the constraints $w_c > 0$, where $c \in \Sigma$ is a constant, and all the constraints $w_g \geq 0$, where $g \in \Sigma$ is a non-constant function symbol.

2. There is a constant $e \in \Sigma$ such that for all constants $c \in \Sigma$ distinct from $e$, $C$ contains the atomic constraint $c \mathrel{?\!\succ} e$.

3. Exactly one of the following conditions holds. (i) There is a unary function symbol $f \in \Sigma$ such that $C$ contains the atomic constraint $w_f \leq 0$, all of the atomic constraints $f \mathrel{?\!\gg} g$ for $g \in \Sigma$ distinct from $f$, and all of the atomic constraints $w_g > 0$ for unary function symbols $g$ distinct from $f$. (ii) For some constant $d \in \Sigma$, $C$ contains the constraint $gtmw(d)$. For every unary function symbol $g \in \Sigma$, $C$ contains the atomic constraint $w_g > 0$.

□

Lemma 7.4.2 *Let $C$ be a rich constraint and the Knuth-Bendix order $\succ$ induced by $(w, \gg)$ satisfies $C$.*

1. *$e$ is the least term with respect to $\succ$.*
2. *There exists a unary function symbol $f \in \Sigma$ such that $w(f) = 0$ if and only if (i) holds. In addition, if such a function $f$ does not exist, then the constraint contains $gtmw(d)$, and hence $d$ is the greatest term of the minimal weight.*
3. *There exists more than one term of the minimal weight if and only if either there exists a unary function symbol $f \in \Sigma$ such that $w(f) = 0$ or there exists a constant $d \in \Sigma$ distinct from $e$ such that $C$ contains the atomic constraint $gtmw(d)$.*

□

Lemma 7.4.3 *The orientability problem can be solved in polynomial time if the orientability problem for rich constraints can be solved in polynomial time.* □

The idea of the proof of the lemma is as follows: one can "guess" the following properties of solutions: (a) which of the constants is smallest one, (b) does there exist a unary function symbol of the weight 0, (c) if such a function does not exist, then which of the constants is the greatest term of the minimal weight. Note that we make only a constant number of guesses.

For technical reasons, we will distinguish two kinds of signatures. Essentially, our algorithm depends on whether the weights of terms are restricted or not. For the *non-trivial* signatures, the weights are not restricted. Note that a signature is non-trivial if and only if it contains either a function symbol of arity $\geq 2$ or at least two function symbols of arity 1. When we present the orientability algorithm for the non-trivial signatures, we will use the fact that terms of sufficiently large weights always exist (see Lemma 6.4.1). A (straightforward) algorithm for trivial signatures is presented in Section 7.6.

## 7.5 The orientability algorithm

In this section we only consider non-trivial signatures. Our algorithm works as follows.

Given a system $R$ of equalities or rewrite rules, we build the initial constraint $C = C_R$. Using Lemma 7.4.3 we can assume that $C$ is rich. We will always denote by $e$ the constant such that $C$ contains all atomic constraints $c \mathrel{?\!\succ} e$, where $c$ is a constant distinct from $e$ (such a constant $e$ exists, since $C$ is rich). Then we repeatedly transform $C$ as described below. We call the *essential size* of a constraint the total number of occurrences of function symbols and variables in its rewriting and orientability part. Every transformation step will either terminate with success or failure, or replace an equality by a rewrite rule, or decrease the essential size of $C$.

At each step the constraint $C$ can be represented as a conjunction $\mathbb{R} \wedge \mathbb{W} \wedge \mathbb{O} \wedge \mathbb{P} \wedge \mathbb{G}$, where $\mathbb{R}$ is a rewrite constraint, $\mathbb{W}$ a weight constraint, $\mathbb{O}$ an orientability constraints, $\mathbb{P}$ a precedence constraint, and $\mathbb{G}$ either empty or has the form $gtmw(c)$.

For every variable $x$ and term $t$, denote by $n(x,t)$ the number of occurrences of $x$ in $t$. For example, $n(x, g(x, h(y, x))) = 2$. Likewise, for every function symbol $g \in \Sigma$ and term $t$, denote by $n(g,t)$ the number of occurrences of $g$ in $t$. For example, $n(h, g(x, h(y, x))) = 1$.

For every term $t$, denote by $W(t)$ the linear expression obtained as follows. Let $v$ be the number of occurrences of variables in $t$. Then

$$W(t) = \sum_{g \in \Sigma} n(g,t) w_g + v w_e. \tag{7.1}$$

For example, if $t = h(x, x, c, e, f(y))$, then

$$W(l) = w_h + w_c + w_f + 4w_e.$$

### 7.5.1 The algorithm

The algorithm works as follows. Every step consists of a number of state transformations, beginning with REWRITE RULE defined below. During the algorithm, we will perform two kinds of *satisfiability checks*:

- The *satisfiability check on* $\mathbb{W}$ is the check whether $\mathbb{W}$ has a solution. If it does not, we terminate with failure.

- The *satisfiability check on* $\mathbb{P}$ is the check whether $\mathbb{P}$ is satisfiable, that is the transitive closure of the set $\{(g,h) \mid g \mathrel{?\!\gg} h$ is an atomic constraint in $\mathbb{P}\}$ is irreflexive. i.e., contains no pair $(g,g)$. If $\mathbb{P}$ is inconsistent, then we terminate with failure.

It is not hard to argue that both kinds of satisfiability checks can be performed in polynomial time. The satisfiability check on $\mathbb{W}$ is polynomial by Lemma 6.2.7. The satisfiability check on $\mathbb{P}$ is polynomial since the transitive closure of a binary relation can be computed in polynomial time, see, e.g. [Cormen et al. 1991].

When any of the sets $\mathbb{W}$ or $\mathbb{P}$ changes, we assume that we perform the corresponding satisfiability check and terminate with failure if it fails.

We will label parts of the algorithm, these labels will be used in the proof of its soundness.

**REWRITE RULE.**

**(R0)** Do the following transformations while possible. If $\mathbb{R}$ contains a tuple inequality $\langle l_1,\dots,l_n\rangle \mathrel{?\!\succ} \langle l_1,\dots,l_n\rangle$, terminate with failure. Otherwise, if $\mathbb{R}$ contains a tuple inequality $\langle l,l_1,\dots,l_n\rangle \mathrel{?\!\succ} \langle l,r_1,\dots,r_n\rangle$, replace it by $\langle l_1,\dots,l_n\rangle \mathrel{?\!\succ} \langle r_1,\dots,r_n\rangle$.

Now $\mathbb{R}$ has the form

$$\begin{gathered}\langle l_1, L_1\rangle \mathrel{?\!\succ} \langle r_1, R_1\rangle,\\ \cdots\\ \langle l_k, L_k\rangle \mathrel{?\!\succ} \langle r_k, R_k\rangle,\end{gathered} \tag{7.2}$$

such that each $l_i$ is a term different from the corresponding term $r_i$.

**(R1)** For all $x$ and $i$ such that $n(x,l_i) > n(x,r_i)$, mark the variable $x$.

**(R2)** If for some $i$ there exists an unmarked variable $x$ such that $n(x,l_i) < n(x,r_i)$, then terminate with failure.

**(R3)** Add to $\mathbb{W}$ all the linear inequalities $W(l_i) \geq W(r_i)$ for all $i$ and perform the satisfiability check on $\mathbb{W}$.

Now compute $\mathbb{W}^=$. If $\mathbb{W}^=$ contains none of the inequalities $W(l_i) \geq W(r_i)$ proceed to EQUALITY. Otherwise, for all $i$ such that $(W(l_i) \geq W(r_i)) \in \mathbb{W}^=$ apply the applicable case below, depending on the form of $l_i$ and $r_i$.

**(R4)** If $l_i = g(s_1, \ldots, s_n)$ and $r_i = h(t_1, \ldots, t_p)$, where $g$ is different from $h$, then replace the constraint $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ by $g \mathrel{?\gg} h$. Perform the satisfiability check on $\mathbb{P}$.

**(R5)** If $l_i = g(s_1, \ldots, s_n)$ and $r_i = g(t_1, \ldots, t_n)$, then replace $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ by $\langle s_1, \ldots, s_n, L_i\rangle \mathrel{?\succ} \langle t_1, \ldots, t_n, R_i\rangle$.

**(R6)** If $(l_i, r_i)$ has the form $(x, y)$, where $x$ and $y$ are different variables, do the following. (Note that at this point both $x$ and $y$ are marked.) If $L_i$ is empty, then terminate with failure. If the constraint guarantees the existence of more than one term of the minimal weight (see Lemma 7.4.2), then also terminate with failure. Otherwise, replace $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ by $\langle L_i\rangle \mathrel{?\succ} \langle R_i\rangle$.

**(R7)** If $(l_i, r_i)$ has the form $(x, t)$, where $t$ is not a variable, do the following. If $t$ is different from $e$, or $L_i$ is empty, then terminate with failure. Otherwise replace in $L_i$ and $R_i$ the variable $x$ by $e$, obtaining $L_i'$ and $R_i'$ respectively, and then replace $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ by $\langle L_i'\rangle \mathrel{?\succ} \langle R_i'\rangle$.

**(R8)** If $(l_i, r_i)$ has the form $(t, x)$, where $t$ is not a variable, do the following. If $t$ contains $x$, remove $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ from $C$. Otherwise, if $t$ is a non-constant or $L_i$ is empty, terminate with failure. (Note that at this point $x$ is marked and $(W(t) \geq W(x)) \in \mathbb{W}^=$.) Let now $t$ be a constant $c$. If $C$ does not contain the atomic constraint $gtmw(c)$, then terminate with failure. Otherwise replace in $L_i$ and $R_i$ the variable $x$ by $c$, obtaining $L_i'$ and $R_i'$ respectively, and then replace $\langle l_i, L_i\rangle \mathrel{?\succ} \langle r_i, R_i\rangle$ by $\langle L_i'\rangle \mathrel{?\succ} \langle R_i'\rangle$.

After this step repeat REWRITE RULE.

**EQUALITY.**

**(E0)** Do the following transformations while possible. If $\mathbb{O}$ contains an atomic constraint $\langle s_1, \ldots, s_n\rangle \mathrel{\prec?\succ} \langle s_1, \ldots, s_n\rangle$, terminate with failure. Otherwise, if $\mathbb{O}$ contains an atomic constraint $\langle s, s_1, \ldots, s_n\rangle \mathrel{\prec?\succ} \langle s, t_1, \ldots, t_n\rangle$, replace it by $\langle s_1, \ldots, s_n\rangle \mathrel{\prec?\succ} \langle t_1, \ldots, t_n\rangle$.

If $\mathbb{O}$ is empty, proceed to TERMINATE. Otherwise, $\mathbb{O}$ now has the form

$$\begin{array}{c} \langle s_1, S_1\rangle \mathrel{\prec?\succ} \langle t_1, T_1\rangle, \\ \cdots \\ \langle s_k, S_k\rangle \mathrel{\prec?\succ} \langle t_k, T_k\rangle, \end{array} \tag{7.3}$$

such that each $s_i$ is a term different from the corresponding term $t_i$.

**(E1)** If, for some $i$ and variable $x$ we have $n(x, s_i) > n(x, t_i)$, then replace $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ by $\langle s_i, S_i\rangle \, ?\succ \langle t_i, T_i\rangle$ and proceed to REWRITE RULE. Likewise, if for some $i$ and variable $x$ we have $n(x, t_i) > n(x, s_i)$, replace $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ by $\langle t_i, T_i\rangle \, ?\succ \langle s_i, S_i\rangle$ and proceed to REWRITE RULE.

Note that after this step for every $i$ and variable $x$, the number of occurrences of $x$ in $s_i$ coincides with its number of occurrences in $t_i$.

Now for each $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ such that $\mathbb{W} \supset W(s_i) = W(t_i)$ apply (E2) below, if there is no such tuples in $\mathbb{O}$ then proceed to TERMINATE.

**(E2)** If the top symbols of $s_i$ and $t_i$ coincide, i.e., we have $s_i = g(u_1, \ldots, u_m)$ and $t_i = g(v_1, \ldots, v_m)$, then we replace the constraint $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ by $\langle u_1, \ldots, u_m, S_i\rangle \prec ?\succ \langle v_1, \ldots, v_m, T_i\rangle$ and proceed to REWRITE RULE. Otherwise, remove $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ from the constraint, and proceed to EQUALITY.

**TERMINATE.** If the constraint contains $gtmw(d)$, then for all constants $c$ different from $d$ such that $w_c \geq w_e$ belongs to $\mathbb{W}^=$ add $d \, ?\gg c$ to the constraint. Perform the satisfiability check on $\mathbb{P}$. Terminate with success.

Note that after TERMINATE, for each $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ either $\mathbb{W} \wedge W(s_i) > W(t_i)$ or $\mathbb{W} \wedge W(t_i) > W(s_i)$ is satisfiable.

### 7.5.2 Correctness

In this section we prove correctness of the algorithm and show how to find a solution when the algorithm terminates with success. The correctness will follow from a series of lemmas asserting that all of the transformation steps performed by the algorithm preserve the set of solutions. Although the algorithm can terminate with success without eliminating all orientability constraints, we will be able to show that in this case the resulting constraint is always satisfiable. To prove this we employ lemmas on homogeneous linear inequalities from Section 7.2.

We will use the following notation and terminology in the proof. We say that a step of the algorithm is *equivalence-preserving* if the set of solutions to the constraint before this step coincides with the set of solutions after the step.

When we use substitutions in the proof, we always assume that the substitutions are grounding for the relevant terms.

The following lemma can be proved by a straightforward application of the definition of solution to a state.

LEMMA 7.5.1 (satisfiability check) *If satisfiability check on* $\mathbb{W}$ *or on* $\mathbb{P}$ *terminates with failure, then* $\mathbb{S}$ *has no solution.* □

In Chapter 6we presented an algorithm for checking orientability of systems of rewrite rules by Knuth-Bendix orders. Since REWRITE RULE uses the same steps as the algorithm in Chapter 6, we can deduce the following lemma about REWRITE RULE.

LEMMA 7.5.2 *Steps (R0)–(R8) are equivalence-preserving.*

PROOF. The proof is the same as for the steps PREPROCESS, (M1)–(M8) of the orientability algorithm for term rewrite rules, see Chapter 6. □

LEMMA 7.5.3 *Step (E1) is equivalence-preserving .*

PROOF. Consider $\langle s_i, S_i\rangle \prec ? \succ \langle t_i, T_i\rangle$ in $\mathbb{O}$ such that for some variable $x$, $n(x, s_i) > n(x, t_i)$. To prove the lemma it suffices to show that if we replace $\langle s_i, S_i\rangle \prec ? \succ \langle t_i, T_i\rangle$ by $\langle t_i, S_i\rangle \, ? \succ \langle s_i, T_i\rangle$ in our constraint, then we obtain an unsatisfiable constraint $C'$. Assume that $C'$ has a solution $\succ$. Let $\sigma$ be any substitution grounding for this tuple inequality. Take any term $u$ and modify $\sigma$ by mapping $x$ into $u$, obtaining $\sigma^u_x$. We have

$$\begin{array}{l} |s_i\sigma^u_x| - |t_i\sigma^u_x| = \\ |s_i\sigma| - |t_i\sigma| + (n(x, s_i) - n(x, t_i)) \cdot (|u| - |x\sigma|). \end{array}$$

Since there exist terms of an arbitrarily large weight, for a term $u$ of a large enough weight we have $|s_i\sigma^u_x| > |t_i\sigma^u_x|$, which contradicts to the assumption $\langle t_i, S_i\rangle\sigma^u_x \succ \langle s_i, T_i\rangle\sigma^u_x$. □

LEMMA 7.5.4 *Step (E2) is equivalence-preserving.*

PROOF. At this step we have that for each variable $x$ the number of occurrences of $x$ in $s_i$ is the same as the the number of occurrences of $x$ in $t_i$ and therefore neither $s_i$ nor $t_i$ is a variable. Also, for every solution to the constraint and every grounding substitution $\sigma$ we have $|s_i\sigma| = |t_i\sigma|$.

Consider the case when top symbols of $s_i$ and $t_i$ coincide, i.e., $s_i = g(u_1, \ldots, u_m)$ and $t_i = g(v_1, \ldots, v_m)$. Then it easy to see that if we have a solution to our constraint such that $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$ the same solution will satisfy the constraint $\langle u_1, \ldots, u_m, S_i\rangle \prec ?\succ \langle v_1, \ldots, v_m, T_i\rangle$ and vice versa.

Now we consider the case when top symbols of $s_i$ and $t_i$ are different, i.e. $s_i = g(\bar{u})$ and $t_i = h(\bar{v})$. It suffices to show that if we have a solution $\succ$ to the constraint after removing $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$, denoted as $C'$, then $\succ$ is also a solution to $\langle s_i, S_i\rangle \prec ?\succ \langle t_i, T_i\rangle$. Consider a solution $\succ$ to $C'$ induced by $(w, \gg)$. Assume that $g \gg h$, then for every substitution $\sigma$ we have $s_i\sigma \succ t_i\sigma$ since $|s_i\sigma| = |t_i\sigma|$. Similar, if $h \gg g$ then for every substitution $\sigma$ we have $t_i\sigma \succ s_i\sigma$. □

Let us show that TERMINATE preserves satisfiability.

LEMMA 7.5.5 TERMINATE *is equivalence-preserving.*

PROOF. Let us show that the addition of all atomic constraints $d \,?\!\gg c$ at this step preserves equivalence. If $C$ has no solution, then this is obvious. Otherwise, take any solution $\succ$ to $C$ and let this solution be induced by $(w, \gg)$. We know $C$ contains $gtmw(d)$, hence $d$ must be the greatest term of the minimal weight. It is not hard to argue that at the TERMINATE step, $\mathbb{W}$ contains all constraints $w_c \geq w_e$, where $c$ is a constant different from $d$. If such a constraint belongs to $\mathbb{W}^=$, then we have $w(c) = w(e)$, hence $c$ is a term of the minimal weight. But then we must have $d \succ c$. By the construction, $C$ also contains $w_e \geq w_d$, so $C \supset w_e = w_d$. Therefore, $d \succ c$ also implies $d \gg c$, and the addition of $d \,?\!\gg c$ does not change the set of solutions. □

We have shown that all steps of our algorithm preserve satisfiability of constraints. Now we show that if the algorithm terminates with success then the constraint is satisfiable, moreover we will be able to find a solution to the constraint in polynomial time.

We call a constraint $C$ *saturated* if application of our orientability algorithm to $C$ does not change $C$ and terminates with success.

LEMMA 7.5.6 *If a constraint* $C = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}$ *is saturated then the constraint* $C' = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G}$ *is satisfiable.*

PROOF. We have that $\mathbb{W}$ is satisfiable, and in addition, for each rewriting constraint $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ the weight constraint $W(l_i) \geq W(r_i)$ does not belong to $\mathbb{W}^=$. By Lemma 6.2.5 there exists a solution $w$ to $\mathbb{W}$ such that for each rewriting constraint $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ we have $W(l_i)w > W(r_i)w$. Let $\succ$ be an order induced by $(w, \gg)$, where $\gg$ is an arbitrary extension of $\mathbb{P}$ to a linear order. We need to show that $\succ$ satisfies the rewriting constraint $\mathbb{R}$ (constraints $\mathbb{W}$,$\mathbb{P}$,$\mathbb{G}$, are obviously satisfied). For this let us consider a tuple $\langle l_i, L_i\rangle \mathrel{?{\succ}} \langle r_i, R_i\rangle$ in $\mathbb{R}$ and an admissible substitution $\sigma$ and show that $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$. From algorithm (rules (R1), (R2)) we have that for each unmarked variable $x$, $n(x, l_i) = n(x, r_i)$, also for each marked variable $y$ we have $|y\sigma| = w(e)$. Therefore

$$|l_i\sigma| - |r_i\sigma| = W(l_i)w - W(r_i)w > 0,$$

this shows that $\langle l_i, L_i\rangle\sigma \succ \langle r_i, R_i\rangle\sigma$. □

LEMMA 7.5.7 *Every saturated constraint is satisfiable.*

PROOF. Consider a saturated constraint

$$C = \mathbb{R} \wedge \mathbb{W} \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}.$$

We show that $C$ is satisfiable by induction on the number of atomic constraints in $\mathbb{O}$. If $\mathbb{O}$ is empty then the claim follows from Lemma 7.5.6. Now assume that $\mathbb{O}$ is not empty. Since $C$ is saturated we have that for each atomic constraint $\langle s_i, S_i\rangle \mathrel{{\prec}?{\succ}} \langle t_i, T_i\rangle$ in $\mathbb{O}$ either $\mathbb{W} \wedge W(s_i) > W(t_i)$ or $\mathbb{W} \wedge W(t_i) > W(s_i)$ is satisfiable. Assume that $\mathbb{W} \wedge W(s_i) > W(t_i)$ is satisfiable, then add $W(s_i) > W(t_i)$ to $\mathbb{W}$ and remove $\langle s_i, S_i\rangle \mathrel{{\prec}?{\succ}} \langle t_i, T_i\rangle$ from $\mathbb{O}$, obtaining $\mathbb{W}'$ and $\mathbb{O}'$ respectively. Let us show that the obtained constraint

$$C' = \mathbb{R} \wedge \mathbb{W}' \wedge \mathbb{P} \wedge \mathbb{G} \wedge \mathbb{O}'$$

is saturated. From Lemma 7.2.1 it follows that the degenerate subsystem of $\mathbb{W}'$ coincides with the degenerate subsystem of $\mathbb{W}$ and since $C$ is saturated we have that none of the rules (R0)–(R8), (E0), (E1) can change the constraint $C'$. Also from Lemma 7.2.4 it follows that for each $\langle s_i', S_i'\rangle \mathrel{{\prec}?{\succ}} \langle t_i', T_i'\rangle$ in $\mathbb{O}'$ either $\mathbb{W}' \wedge W(s_i') > W(t_i')$ or $\mathbb{W}' \wedge W(t_i') > W(s_i')$ is satisfiable. Hence, rule (E2) also can not change the constraint $C'$ and we conclude that $C'$ is saturated. Since $\mathbb{O}'$

contains less atomic constraints than $\mathbb{O}'$ and $C'$ is saturated, we can apply the induction hypothesis.

$\square$

### 7.5.3 Time complexity

Provided that we use a polynomial-time algorithm for solving systems of homogeneous linear inequalities, and a polynomial-time algorithm for transitive closure, a careful analysis of our algorithm shows the following.

LEMMA 7.5.8 *The algorithm runs in time polynomial of the size of the system of rewrite rules.* $\square$

## 7.6 Orientability for trivial signatures

In this section we consider only trivial signatures. Let us remind that a signature is called *trivial* if it contains no function symbols of arity $\geq 2$, and at most one unary function symbol. Consider a trivial signature which consists of a unary function symbol $g$ and some constants. Consider a constraint $C = \mathbb{R} \wedge \mathbb{O}$ where $\mathbb{R}$ is a rewriting constraint and $\mathbb{O}$ is an orientability constraint. If $\mathbb{O}$ contains an orientability constraint $t \prec?\succ t$ then $C$ is obviously unsatisfiable, and therefore we will assume that for all orientability constraints $t \prec?\succ s \in \mathbb{O}$ , $t$ is different from $s$. If $\mathbb{O}$ contains nonground constraints then we can transform $C$ into an equally orientable constraint $C' = \mathbb{R}' \wedge \mathbb{O}'$ such that all constraints in $\mathbb{O}'$ are ground.

LEMMA 7.6.1 *Let $C = \mathbb{R} \wedge \mathbb{O}$ be a constraint in a trivial signature $\Sigma$ such that $\mathbb{O}$ contains nonground atomic constraints. Define a constraint $C' = \mathbb{R}' \wedge \mathbb{O}'$ obtained by the following transformations.*

1. *Replace every atomic orientability constraint $g^m(x) \prec?\succ g^n(d)$ with the rewriting constraint $g^m(x) ?\succ g^n(d)$.*
2. *Replace every atomic orientability constraint $g^m(x) \prec?\succ g^n(x)$, where $m > n$, with the rewriting constraint $g^m(x) ?\succ g^n(x)$.*

*Then a Knuth-Bendix order $\succ$ orients $C$ if and only if it orients $C'$. All constraints in $\mathbb{O}'$ are ground.* $\square$

The proof of this lemma is straightforward and we can restrict ourselves to constraints with ground orientability part. Now, consider such a constraint $C = \mathbb{R} \wedge \mathbb{O}$ and let $\succ$ be a Knuth-Bendix order which is a solution to $\mathbb{R}$. Then $\succ$ is also a solution to $\mathbb{O}$, since every Knuth-Bendix order is total on ground terms. Therefore, we reduce our problem to the problem of orientability of rewriting systems for trivial signatures which is shown to be solvable in polynomial time in Section 6.6.

## 7.7 Main results

Lemmas 7.5.1–7.5.7 guarantee that the orientability algorithm is correct and Lemma 7.5.8 implies that it runs in polynomial time. Hence we obtain the following theorem.

THEOREM 7.7.1 *The problem of the existence of a Knuth-Bendix order which orients a given system of equalities and rewrite rules can be solved in the time polynomial in the size of the system. Moreover, if the system of equalities and rewrite rules is orientable by a Knuth-Bendix order we can find such an order in polynomial time.* □

In Chapter 6 we have proved that the problem of orientability by Knuth-Bendix orders is P-complete for systems of rewrite rules, moreover it is P-hard even for ground rewrite rule systems. Therefore, the following result follows from Theorem 6.9.2 and Proposition 7.1.2.

THEOREM 7.7.2 *The problem of orientability of systems of equalities and rewrite rules by Knuth-Bendix orders is P-complete. Moreover, it is P-hard even for systems consisting only of equalities or only of rewrite rules.* □

# Chapter 8

# AC-Compatible Knuth-Bendix Orders

## 8.1 Introduction

This chapter is based on the paper [Korovin and Voronkov 2003$a$].

$E$-compatible simplification orders for various equational theories $E$ can be used for building-in equational theories in theorem provers and rewriting modulo equational theories (see Chapter 2).

Among various equational theories, theories axiomatized by the axioms of associativity and commutativity, so-called AC-theories, play a special role. Such theories very often occur in applications and require special treatment in automated systems, where AC-compatible simplification orders is a crucial ingredient.

The existence of an AC-compatible simplification order AC-total on ground terms had been a challenging problem for many years, which was finally solved in [Narendran and Rusinowitch 1991]. Recently, a lot of work has been done to modify recursive path orders to obtain AC-compatible simplification orders AC-total on ground terms [Rubio and Nieuwenhuis 1993, Rubio 2002, Rubio 1999, Kapur and Sivakumar 1998, Kapur and Sivakumar 1997, Kapur et al. 1995, Kapur et al. 1990]. Despite the fact that Knuth-Bendix orders are widely used in automated deduction, to our knowledge no AC-compatible simplification variant of Knuth-Bendix orders have been known. (There was an attempt to introduce such an order in [Steinbach 1990] but this order is lacking the crucial monotonicity property, as we will show later).

In this chapter we define a family of AC-compatible Knuth-Bendix orders

$\succ_{\texttt{ACKBO}}$. These orders enjoy attractive features of the standard Knuth-Bendix orders, for example

1. a polynomial-time algorithm for term comparison;
2. computationally efficient approximations based on weight comparison, so in many practical cases we do not need to traverse the whole term each time to compare it with another term;
3. light terms are smaller than heavier ones.

Our approach share some ideas with the AC-RPO of Rubio [Rubio 2002, Rubio 1999], but a careful exploitation of some properties of weight functions enable us to avoid complications leading to an exponential behavior in the AC-RPO case.

## 8.2 Preliminaries

We will use multisets and multiset extension of an order, as defined in Chapter 3, where key properties of such extensions are discussed.

DEFINITION 8.2.1 Let $>$ be a binary relation on a set $S$. A *lexicographic extension* of $>$, denoted by $>^{lex}$, is a relation on tuples of elements of $S$ defined as follows. Let $\bar{a} = (a_1, \ldots, a_m)$ and $\bar{b} = (b_1, \ldots, b_n)$ be two tuples. Then $\bar{a} >^{lex} \bar{b}$ if one of the following conditions holds:

1. $m > n$;
2. $m = n$ and there exists $i$ such that $1 \leq i \leq m$, $a_i > b_i$, and for all $j \in \{1, \ldots, i-1\}$ we have $a_j = b_j$.

□

The following fact is not hard to check, see, e.g., [Baader and Nipkow 1998].

LEMMA 8.2.2 *If $>$ is an order, then so is $>^{lex}$. If $>$ is a total order, then so is $>^{lex}$. If $>$ is a well-founded order, then so is $>^{lex}$.* □

For every pre-order $\geq$ we denote by $>$ the corresponding strict order $>$ defined as follows: $s > t$ if and only if $s \geq t$ and $t \not\geq s$. We will use this notation for various pre-orders, for example $\succ$ will denote the strict version of $\succeq$.

Let $\geq_1, \geq_2$ be pre-orders. We call the *lexicographic product* of $\geq_1$ and $\geq_2$, denoted $\geq_1 \otimes \geq_2$, the relation $\geq$ defined as follows: $s \geq t$ if and only if either $s >_1 t$, or $s \geq_1 t$ and $s \geq_2 t$. It is not hard to argue that $\geq_1 \otimes \geq_2$ is a pre-order. We define lexicographic product $>_1 \otimes >_2$ of strict parts of $\geq_1, \geq_2$ as the strict part of $\geq_1 \otimes \geq_2$.

We will also consider lexicographic products of more than two orders.

LEMMA 8.2.3 *If $>_1, >_2$ are orders, then so is $>_1 \otimes >_2$. If $>_1, >_2$ are total orders, then so is $>_1 \otimes >_2$. If $>_1, >_2$ are well-founded orders, then so is $>_1 \otimes >_2$.* □

In our proofs below we will often compose the multiset order, the lexicographic extension, and the lexicographic product of various orders and use Lemmas 3.2.4, 8.2.2 and 8.2.3 to establish properties of the compositions.

## 8.3 AC-compatible orders

Let $E$ be an equational theory and $>$ be a partial order on ground terms of a signature $\Sigma$. Denote equality with respect to $E$ by $=_E$. We say that an order $>$ is *$E$-compatible* if it satisfies the following property: if $s > t$, $s =_E s'$ and $t =_E t'$, then $s' > t'$. The order $>$ is called *$E$-total*, if for all ground terms $s, t$, if $s \neq_E t$, then either $s > t$ or $t > s$.

Let $+$ be a binary function symbol. The *AC-theory* for $+$ is the equational theory axiomatized by set of two formulas

$$\forall x \forall y \forall z((x + y) + z \simeq x + (y + z));$$
$$\forall x \forall y(x + y \simeq y + x).$$

From now on we assume that we are given a fixed signature $\Sigma$ with a distinguished subset $\Sigma_{AC}$ of binary function symbols. The members of $\Sigma_{AC}$ will be called *AC-symbols*. Two terms $s, t$ are called *AC-equal*, denoted $s =_{AC} t$, if they are equal in the equational theory generated by the union of the AC-theories for all $g \in \Sigma_{AC}$. An order is called *AC-compatible* if it is $E$-compatible with respect to this equational theory.

## 8.4 Main results

Our main aim is to find AC-compatible AC-total simplification orders which generalize standard Knuth-Bendix orders for the case of AC-theories. In the rest of this chapter we define a family of such orders, each order $\succ_{\texttt{ACKBO}}$ in this family is induced by a weight function $w$ and a precedence relation $\gg$ compatible with $w$. We prove the following results.

1. $\succ_{\texttt{ACKBO}}$ is an AC-compatible AC-total simplification order,
2. On the terms without AC-symbols, $\succ_{\texttt{ACKBO}}$ coincides with the standard Knuth-Bendix order induced by $w$ and $\gg$.
3. If $\Sigma$ contains no unary function symbols of the weight 0, then for every ground term $t$ there exists a finite number of terms $s$ such that $t \succ_{\texttt{ACKBO}} s$.

Further, we extend the orders $\succ_{\texttt{ACKBO}}$ to non-ground terms in such a way that for all terms $s, t$ and substitutions $\theta$, if $s \succ_{\texttt{ACKBO}} t$, then $s\theta \succ_{\texttt{ACKBO}} t\theta$.

## 8.5 The Ground Case

### 8.5.1 Flattened terms

In the sequel the symbol $+$ will range over $\Sigma_{AC}$. Let us call a term *normalized* if it has no subterms of the form $(r+s)+t$. Evidently, every term is AC-equal to a normalized term. Since we aim at finding AC-compatible simplification orders, it is enough for us to define these orders only for normalized terms. For normalized terms, we introduce a special well-known notation, called *flattened term*.

To this end, we consider all AC-symbols to be varyadic, i.e., having an unbounded arity greater than or equal to 2. A term $s$ using the varyadic symbols is called *flattened* if for every non-variable subterm $t$ of $s$, if $t$ has the form $+(t_1, \ldots, t_n)$, then the top symbols of $t_1, \ldots, t_n$ are distinct from $+$. We identify a subterm $+(t_1, \ldots, t_m)$ with the normalized term $(t_1 + (t_2 + \ldots + t_n))$. We will sometime write subterms of flattened terms as $t_1 + \ldots + t_n$. In the sequel we will only deal with flattened terms.

Note that we have to be careful with defining substitutions into flattened terms and the subterm property for them. When we substitute a term $s_1 + \ldots + s_m$ for a variable $x$ in $x + t_1 + \ldots + t_n$, we obtain $s_1 + \ldots + s_m + t_1 + \ldots + t_n$. To prove the subterm property for an order $>$ on ordinary terms, we also have to prove the following *cancellation property* for flattened terms: $s_1+s_2+\ldots+s_n > s_2+\ldots+s_n$.

Similarly, we have to be careful with defining weights of terms with varyadic symbols. We want the weight to be invariant under $=_{AC}$, in particular, the weight of a term must coincide with the weight of a flattened term equal to it modulo AC. Therefore, we modify the definition of weight as follows.

Definition 8.5.1 (Weight) The *weight* of a ground term $t$, denoted $|t|$, is defined as follows. Let $t = g(t_1, \ldots, t_n)$, where $n \geq 0$. Then

1. if $g \notin \Sigma_{AC}$, then $|t| = w(g) + |t_1| + \ldots + |t_n|$.
2. if $g \in \Sigma_{AC}$, then $|t| = (n-1)w(g) + |t_1| + \ldots + |t_n|$.

$\square$

We have the following straightforward result.

Lemma 8.5.2 *Let $r, s, t$ be terms. If $|s| = |t|$, then $|r[s]| = |r[t]|$. Likewise, if $|s| > |t|$, then $|r[s]| > |r[t]|$.* $\square$

### 8.5.2 Relation $\succ_+$

All relations introduced below will be AC-compatible. Therefore, in the sequel we will consider the AC-equality instead of the syntactic equality and consider relations on the equivalence classes modulo $=_{AC}$.

To define an AC-compatible weight-based simplification order, let us first define, for each AC-symbol $+$, an auxiliary partial order $\succ_+$ on multisets of flattened terms.

If a term $t$ has the form $g(t_1, \ldots, t_n)$, where $n \geq 0$, then $g$ is called the *top symbol* of $t$, denoted by $top(t)$, and $t_1, \ldots, t_n$ the *arguments* of $t$. We define the top symbol of a variable $x$ to be $x$ itself.

First we introduce the following *pre-order* $\geq_{top}$ on terms: $s \geq_{top} t$ if and only if $top(s) \gg top(t)$ or $top(s) = top(t)$. Note that this order is also defined for non-ground terms. Likewise, we introduce the pre-order $\geq_w$ on ground terms as follows: $s \geq_w t$ if $|s| \geq |t|$. Naturally, the strict versions of $\geq_{top}$ and $\geq_w$ are denoted by $>_{top}$ and $>_w$, respectively.

DEFINITION 8.5.3 (Relation $\succ_+$) Let $M, N$ be two multisets of flattened ground terms and let

$$M' = \dot{\{} t \dot{\in} M \mid top(t) \gg + \dot{\}};$$
$$N' = \dot{\{} t \dot{\in} N \mid top(t) \gg + \dot{\}}.$$

We define $M \succeq_+ N$ if and only if

$$M' \ (\geq_w \otimes \geq_{top})^{mul} \ N'.$$

□

In other words, we can define the order $\succ_+$ as follows. First, remove from $M$ and $N$ all elements with top symbols smaller than or equal to $+$. Then compare the remaining multisets using the multiset order in which the terms are first compared by weight and then by their top symbol.

LEMMA 8.5.4 *For each symbol* $+ \in \Sigma_{AC}$ *the relation* $\succ_+$ *is a well-founded order.*

PROOF. Follows immediately from the observation that the strict part of $(\geq_w \otimes \geq_{top})^{mul}$ is a well-founded order (by Lemmas 3.2.4 and 8.2.3). □

Let us give a characterization of the relation $\succ_+$. Let $M$ be a multiset of ground terms and $v$ be a positive integer. Denote by $selected(+, v, M)$ the multiset of top functors of all terms in $M$ of the weight $v$ whose top symbol is greater than $+$ w.r.t. $\gg$. Then we have $M \succ_+ N$ if and only if there exists an integer $v$ such that $selected(+, v, M) >_{top}^{mul} selected(+, v, N)$ and for all $v' > v$, $selected(+, v', M) =^{mul} selected(+, v', N)$. Let $\equiv_+$ denote the incomparability relation on multisets of terms w.r.t. $\succ_+$. That is, given two multisets $M, N$, we have $M \equiv_+ N$ if and only if neither $M \succ_+ N$ nor $N \succ_+ M$. Now it is easy to check that two multisets of terms $M$ and $N$ are incomparable w.r.t. $\succ_+$ if and only if for each weight $v$ we have $selected(+, v, M) = selected(+, v, N)$ and therefore $\equiv_+$ is indeed an equivalence relation on terms. So $\succ_+$ can be seen as a total well-founded order on the equivalence classes of multisets modulo $\equiv_+$.

### 8.5.3 Order $\succ_{\texttt{ACKBO}}$

Using the relation $\succ_+$, we can define an AC-compatible simplification order $\succ_{\texttt{ACKBO}}$.

DEFINITION 8.5.5 (Order $\succ_{\mathtt{ACKBO}}$) Let $t = h(t_1, \ldots, t_n)$ and $s = g(s_1, \ldots, s_k)$ be flattened ground terms. Then $t \succ_{\mathtt{ACKBO}} s$ if and only if one of the following conditions holds:

1. $|t| > |s|$; or
2. $|t| = |s|$ and $h \gg g$; or
3. $|t| = |s|$, $h = g$, and either
   (a) $h \notin \Sigma_{AC}$ and $(t_1, \ldots, t_n) \succ^{lex}_{\mathtt{ACKBO}} (s_1, \ldots, s_n)$; or
   (b) $h \in \Sigma_{AC}$ and
      i. $\dot{\{}t_1, \ldots, t_n\dot{\}} \succ_h \dot{\{}s_1, \ldots, s_k\dot{\}}$; or
      ii. $\dot{\{}t_1, \ldots, t_n\dot{\}} \equiv_h \dot{\{}s_1, \ldots, s_k\dot{\}}$ and $n > k$; or
      iii. $\dot{\{}t_1, \ldots, t_n\dot{\}} \equiv_h \dot{\{}s_1, \ldots, s_k\dot{\}}$, $n = k$ and $\dot{\{}t_1, \ldots, t_n\dot{\}} \succ^{mul}_{\mathtt{ACKBO}} \dot{\{}s_1, \ldots, s_k\dot{\}}$.

□

Let us remark that similar to the AC-RPO of Rubio [Rubio 2002, Rubio 1999] we make a special treatment of the immediate subterms below $+$ having top symbols greater than $+$. To this end, we use the relation $\succ_+$, which allows us to avoid recursive computations deeper into subterms at this stage (we need only to compare weights and top symbols of the immediate subterms). As a result, we gain some efficiency. More importantly, using properties of the weight functions we can avoid the exponential behavior of AC-RPO caused by enumerating embeddings of certain subterms.

LEMMA 8.5.6 *$\succ_{\mathtt{ACKBO}}$ is an AC-compatible AC-total order on ground terms.*

PROOF. It is easy to see that $\succ_{\mathtt{ACKBO}}$ is AC-compatible. The AC-totality can be proved by a routine induction on terms.

Let us prove that $\succ_{\mathtt{ACKBO}}$ is an order. Let us call the *$f$-height* of a term $r$, denoted by $height_f(r)$, the greatest number $n$ such that $r = f^n(r')$. The proof is by induction on the order $>'$ on ground terms defined as follows: $t >' s$ if $|t| > |s|$ or $|t| = |s|$ and $height_f(t) > height_f(s)$. Obviously, $>'$ is the lexicographic product of two well-founded orders, and so a well-founded order itself.

Note the following property of $>'$: if $t >' s$, then $t \succ_{\mathtt{ACKBO}} s$. Therefore, it is enough to prove that for each pair of natural numbers $(k, l)$, the relation $\succ_{\mathtt{ACKBO}}$ is an order on the set of ground terms

$$\{t \mid |t| = k \text{ and } height_f(t) = l\}.$$

But this follows from the following observation: $\succ_{\mathtt{ACKBO}}$ on this set of terms is defined as a lexicographic product of the following five orders:

$$
\begin{aligned}
& t >_1 s \Leftrightarrow h \gg g; \\
& t >_2 s \Leftrightarrow (t_1, \ldots, t_n) \succ^{lex}_{\mathtt{ACKBO}} (s_1, \ldots, s_n) \text{ and } h = g \notin \Sigma_{AC}; \\
& t >_3 s \Leftrightarrow \dot{\{}t_1, \ldots, t_n\dot{\}} \succ_h \dot{\{}s_1, \ldots, s_k\dot{\}} \text{ and } h = g \in \Sigma_{AC}; \\
& t >_4 s \Leftrightarrow n > k \text{ and } h = g \in \Sigma_{AC}; \\
& t >_5 s \Leftrightarrow \dot{\{}t_1, \ldots, t_n\dot{\}} \succ^{mul}_{\mathtt{ACKBO}} \dot{\{}s_1, \ldots, s_k\dot{\}} \text{ and } h = g \in \Sigma_{AC}.
\end{aligned}
$$

Note that $\succ^{lex}_{\mathtt{ACKBO}}$ and $\succ^{mul}_{\mathtt{ACKBO}}$ used in this definition are orders by the induction hypothesis and by Lemmas 8.2.2 and 3.2.4. $\square$

THEOREM 8.5.7 *The relation $\succ_{\mathtt{ACKBO}}$ is an AC-compatible AC-total simplification order on ground terms.*

PROOF. By Lemma 8.5.6, $\succ_{\mathtt{ACKBO}}$ is an order, so it only remains to prove the subterm property, cancellation property, and monotonicity. The cancellation property is obvious, since $|s_0 + s_1 + \ldots + s_n| > |s_1 + \ldots + s_n|$. The subterm property is checked in the same way as for the standard Knuth-Bendix order.

Let us prove the monotonicity. By Lemma 8.5.6, $\succ_{\mathtt{ACKBO}}$ is an AC-compatible AC-total order. In particular, $\succ_{\mathtt{ACKBO}}$ is transitive, so it remains to prove the following property: if $t \succ_{\mathtt{ACKBO}} s$, then for every function symbol $g$ we have $g(r_1, \ldots, r_{i-1}, t, r_{i+1}, \ldots, r_n) \succ g(r_1, \ldots, r_{i-1}, s, r_{i+1}, \ldots, r_n)$. When $g \notin \Sigma_{AC}$, the proof is identical to that for the standard Knuth-Bendix order, so we only consider the case when $g$ is an AC-symbol $+$.

We have to prove the following statement for all terms $s, t, r_1, \ldots, r_m$: let $u = t + r_1 + \ldots + r_m$ and $v = s + r_1 + \ldots + r_m$, then $t \succ_{\mathtt{ACKBO}} s$ implies $u \succ_{\mathtt{ACKBO}} v$. Let $t = h(t_1, \ldots, t_n)$ and $s = g(s_1, \ldots, s_k)$. Consider all possible cases of Definition 8.5.5 of $\succ_{\mathtt{ACKBO}}$.

1. $|t| > |s|$. In this case by Lemma 8.5.2 we have $|u| > |v|$, and so $u \succ_{\mathtt{ACKBO}} v$.

Now we can assume $|t| = |s|$, hence by Lemma 8.5.2 $|u| = |v|$. Denote by $U$ and $V$ the multisets of arguments of $u$ and $v$, respectively. Note that $U$ is not necessarily equal to $\dot{\{}t, r_1, \ldots, r_m\dot{\}}$: indeed, the top symbol of $t$ may be $+$, and then we have to flatten $t + r_1 + \ldots + r_m$ to obtain the arguments of $u$. Likewise, $V$ is not necessarily equal to $\dot{\{}s, r_1, \ldots, r_m\dot{\}}$. Denote by $p, q$ the number of elements in $U, V$ respectively. Note that

$$p = \begin{cases} m+1, & \text{if } top(t) \neq +; \\ m+n, & \text{if } top(t) = +. \end{cases}$$
$$q = \begin{cases} m+1, & \text{if } top(s) \neq +; \\ m+k, & \text{if } top(s) = +. \end{cases}$$

Since $|u| = |v|$ and $top(u) = top(v) = +$, the comparison of $u$ and $v$ should be done using clauses (3(b)i)–(3(b)iii) of Definition 8.5.5. That is, first we check $U \succ_+ V$. Then, if $U \equiv_+ V$, we check if $p > q$. Finally, if $p = q$, we compare $U$ and $V$ using the multiset order $\succ^{mul}_{\mathtt{ACKBO}}$. Consider the remaining cases.

2. $h \gg g$. Let us show that if $h \gg +$ then $U \succ_+ V$ and so $u \succ_{\mathtt{ACKBO}} v$. If $+ \gg g$ then we have $U \succ_+ U \dot{-} \dot{\{}t\dot{\}} = \dot{\{}r_1, \ldots, r_m\dot{\}} = V \dot{-} \dot{\{}s\dot{\}} \equiv_+ V$. If $g \gg +$ then $\dot{\{}t\dot{\}} \succ_+ \dot{\{}s\dot{\}}$ and hence $U = \dot{\{}t, r_1, \ldots, r_m\dot{\}} \succ_+ \dot{\{}s, r_1, \ldots, r_m\dot{\}} = V$. If $g = +$ then $s$ is of the form $s_1 + \ldots + s_k$. We have $\dot{\{}t\dot{\}} \succ_+ \dot{\{}s_1, \ldots, s_k\dot{\}}$, since the weight of each arguments of $s$ is strictly less than the weight of $t$, and therefore $U \succ_+ V$.

   Now if $+ \gg h$, then $U \equiv_+ V$ and $p = q$. In this case $u \succ_{\mathtt{ACKBO}} v \Leftrightarrow U \succ^{mul}_{\mathtt{ACKBO}} V \Leftrightarrow t \succ_{\mathtt{ACKBO}} s$, so $u \succ_{\mathtt{ACKBO}} v$. It remains to consider the case $h = +$. In this case we have $U \succeq_+ V \dot{-} \dot{\{}s\dot{\}} \equiv_+ V$ and either $U \succ_+ V$, so $u \succ_{\mathtt{ACKBO}} v$, or we have $U \equiv_+ V$ and $p > q$, so $u \succ_{\mathtt{ACKBO}} v$, by (3(b)ii) of Definition 8.5.5.

3. $h = g$.

   (a) $h \neq +$. Then $U \equiv_+ V$ and $p = q$. In this case $u \succ_{\mathtt{ACKBO}} v \Leftrightarrow U \succ^{mul}_{\mathtt{ACKBO}} V \Leftrightarrow t \succ_{\mathtt{ACKBO}} s$.

   (b) Now it remains to consider the case $h = g = +$. In this case $U = \dot{\{}t_1, \ldots, t_n, r_1, \ldots, r_m\dot{\}}$ and $V = \dot{\{}s_1, \ldots, s_k, r_1, \ldots, r_m\dot{\}}$. Since $t \succ_{\mathtt{ACKBO}} s$, it is enough to consider the following cases.

i. $\dot{\{}t_1, \ldots, t_n\dot{\}} \succ_+ \dot{\{}s_1, \ldots, s_k\dot{\}}$. In this case $U \succ_+ V$, hence $u \succ_{\mathtt{ACKBO}} v$.

ii. $\dot{\{}t_1, \ldots, t_n\dot{\}} \equiv_+ \dot{\{}s_1, \ldots, s_k\dot{\}}$ and $n > k$. In this case $U \equiv_+ V$ but $p > q$, hence $u \succ_{\mathtt{ACKBO}} v$.

iii. $\dot{\{}t_1, \ldots, t_n\dot{\}} \equiv_+ \dot{\{}s_1, \ldots, s_k\dot{\}}$, $n = k$, and $\dot{\{}t_1, \ldots, t_n\dot{\}} \succ^{mul}_{\mathtt{ACKBO}} \dot{\{}s_1, \ldots, s_k\dot{\}}$. In this case $U \equiv_+ V$, $p = q$, but $U \succ^{mul}_{\mathtt{ACKBO}} V$, hence $u \succ_{\mathtt{ACKBO}} v$.

The proof is complete. $\square$

Suppose that $\Sigma$ does not contains a unary function symbol $f$ of the weight 0. In this case for each weight $v$ there is only a finite number of ground terms of the weight $v$. Therefore, we have the following result.

PROPOSITION 8.5.8 *If $\Sigma$ does not contain a unary function symbol $f$ of the weight* 0*, then for every term $t$, there exists only a finite number of terms $s$ such that $t \succ_{\mathtt{ACKBO}} s$.* $\square$

Now let us show that if our signature contains only two AC-symbols and in addition one of them is maximal and another is minimal w.r.t. $\gg$, then we can considerably simplify definition of AC-KBO by avoiding $\succ_h$ comparisons. In particular the following definition will satisfy all required properties.

DEFINITION 8.5.9 (Simplified AC-KBO for two AC symbols) Consider a signature $\Sigma$ containing only two AC-symbols, such that one of them is maximal and another is minimal w.r.t. $\gg$ in $\Sigma$.

Let $t = h(t_1, \ldots, t_n)$ and $s = g(s_1, \ldots, s_k)$ be flattened ground terms. Then $t \succ'_{\mathtt{ACKBO}} s$ if and only if one of the following conditions holds:

1. $|t| > |s|$; or

2. $|t| = |s|$ and $h \gg g$; or

3. $|t| = |s|$, $h = g$, and either

   (a) $h \notin \Sigma_{AC}$ and $(t_1, \ldots, t_n) \succ'^{\,lex}_{\mathtt{ACKBO}} (s_1, \ldots, s_n)$; or

   (b) $h \in \Sigma_{AC}$ and

      i. $n > k$ and $h$ is maximal in $\Sigma$ w.r.t. $\gg$; or

ii. $k > n$ and $h$ is minimal in $\Sigma$ w.r.t. $\gg$; or

iii. $k = n$ and $\dot{\{}t_1, \ldots, t_n\dot{\}} \succ'^{mul}_{\mathtt{ACKBO}} \dot{\{}s_1, \ldots, s_k\dot{\}}$.

□

THEOREM 8.5.10 *The relation $\succ'_{\mathtt{ACKBO}}$ is an AC-compatible AC-total simplification order on ground terms.*

PROOF. We skip the proof which is similar to the general case. □

## 8.6 Non-Ground Order

In this section we will generalize AC-compatible Knuth-Bendix orders $\succ_{\mathtt{ACKBO}}$ to non-ground terms. The definition will be very similar to the ground case. We will have to change the definitions of the weight and slightly change the definition of $\succ_+$. As before, we will be dealing with flattened terms.

Let us modify the notion of weight to non-ground terms. In fact, we will introduce two different weights $|t|$ and $||t||$. As before, we assume that we are given a weight function $w$ and a precedence relation $\gg$ compatible with $w$. Let $e$ denote the constant in $\Sigma$ having the least weight among all constants in $\Sigma$. It is not hard to argue that $|e|$ is also the least weight of a ground term.

DEFINITION 8.6.1 (Weight $|t|$) The *weight* of a term $t$, denoted $|t|$, is defined as follows.

1. If $t$ is a variable, then $|t| = w(e)$.
2. If $t = g(t_1, \ldots, t_n)$ and $g \notin \Sigma_{AC}$, then $|t| = w(g) + |t_1| + \ldots + |t_n|$.
3. If $t = g(t_1, \ldots, t_n)$ and $g \in \Sigma_{AC}$, then $|t| = (n-1)w(g) + |t_1| + \ldots + |t_n|$.

□

It is not hard to argue that the weight of a term $t$ is equal to the weight of the ground term obtained from $t$ by replacing all variables by $e$. Therefore, Lemma 8.5.2 also holds for non-ground terms.

LEMMA 8.6.2 *Let $r, s, t$ be terms. If $|s| = |t|$, then $|r[s]| = |r[t]|$. Likewise, if $|s| > |t|$, then $|r[s]| > |r[t]|$.* □

Let $t$ be a term. Denote by $vars(t)$ the multiset of variables of $t$. For example, $vars(g(x, a, h(y, x))) = \dot{\{}x, y, x\dot{\}}$.

DEFINITION 8.6.3 (Generalized Weight) A *generalized weight* is a pair $(n, V)$, where $n$ is a positive integer and $V$ is a multiset of variables. Let us introduce a pre-order $\geq$ and an order $>$ on generalized weights as follows. We let $(m, M) \geq (n, N)$ if $m \geq n$ and $N$ is a submultiset of $M$. We let $(m, M) > (n, N)$ if $m > n$ and $N$ is a submultiset of $M$. The *generalized weight of a term* $t$, denoted $||t||$, is the pair $(|t|, vars(t))$. We write $t \geq_W s$ if $||t|| \geq ||s||$ and $t >_W s$ if $||t|| > ||s||$. □

Note that $>_W$ is *not* a strict version of $\geq_W$. However, it is easy to see that $>_W$ is a well-founded order. The following properties of $\geq_W$ and $>_W$ are also not difficult to check.

LEMMA 8.6.4 *Let $r, s, t$ be terms. If $s \geq_W t$, then $r[s] \geq_W r[t]$. Likewise, if $s >_W t$, then $r[s] >_W r[t]$. Moreover, if $s, t$ are ground terms, then $s \geq_w t$ if and only if $s \geq_W t$, and $s >_w t$ if and only if $s >_W t$.* □

Note that $\geq_W$ is not a total pre-order. For example, if $x, y$ are two different variables, then neither $x \geq_W y$ nor $y \geq_W x$ holds.

### 8.6.1 Relation $\succ_+$

Let us now generalize the relation $\succ_+$ to non-ground terms. The definition is more complex that in the ground case because of one technical problem: the order $>_W$ is not the strict version of $\geq_W$. Therefore, we cannot compose orders using $\geq_W$ to obtain new orders as before. In particular, the definition of a multiset extension of an order does not work any more and should be replaced.

First, instead of the pre-order $\geq_w \otimes \geq_{top}$ used in the definition of $\succ_+$ on ground terms, we introduce a pre-order $\geq_{W,top}$ defined as $\geq_W \otimes \geq_{top}$. We also write $s =_{W,top} t$ if $||s|| = ||t||$ and $top(s) = top(t)$. Then let us define an order $>_{W,top}$ as follows: $s >_{W,top} t$ if either $s >_W t$ or $s \geq_W t$ and $top(s) \gg top(t)$.

Now, to define an analogue of $(\geq_w \otimes \geq_{top})^{mul}$ used in the definition of $\succ_+$ for ground terms, let us define the following *deletion operation* on pairs of multisets $M, N$: if $t \mathrel{\dot\in} M$, $s \mathrel{\dot\in} N$, and $t =_{W,top} s$, then delete one occurrence of $t$ from $M$ and one occurrence of $s$ from $N$.

DEFINITION 8.6.5 (Relation $\succ_+$) Let $M, N$ be two multisets of flattened terms and let

$$M' = \dot{\{} t \mathrel{\dot{\in}} M \mid t \text{ is a variable or } top(t) \gg + \dot{\}};$$
$$N' = \dot{\{} t \mathrel{\dot{\in}} N \mid t \text{ is a variable or } top(t) \gg + \dot{\}}.$$

Let $M'', N''$ be obtained by applying the deletion operation to $M', N'$ while possible. Then we define $M \succ_+ N$ if $M''$ contains a non-variable term and for every $s \in N''$ there exists $t \in M''$ such that $t >_{W,top} s$. We also define $M \succeq_+ N$ if either $M \succ_+ N$ or $N''$ is empty and $M''$ contains only variables. □

Similarly to the ground case, we have the following lemma.

LEMMA 8.6.6 *For each symbol $+ \in \Sigma_{AC}$ the relation $\succ_+$ is a well-founded order. Moreover, on ground terms it coincides with the order $\succ_+$ of Definition 8.5.3.* □

### 8.6.2 Order $\succ_{\mathtt{ACKBO}}$

Using the relation $\succ_+$, we can define an AC-compatible simplification order $\succ_{\mathtt{ACKBO}}$ in essentially the same way as for ground terms.

DEFINITION 8.6.7 (Order $\succ_{\mathtt{ACKBO}}$) Let us define the relation $\succ_{\mathtt{ACKBO}}$ for non-ground terms as follows. If $x$ is a variable, then for every term $s$ it is not true that $x \succ_{\mathtt{ACKBO}} s$. If $y$ is a variable then $t \succ_{\mathtt{ACKBO}} y$ if and only if $y$ occurs in $t$ and is distinct from $t$. Let $t = h(t_1, \dots, t_n)$ and $s = g(s_1, \dots, s_k)$ be flattened terms. Then $t \succ_{\mathtt{ACKBO}} s$ if and only if one of the following conditions holds:

1. $t >_W s$; or
2. $t \geq_W s$ and $h \gg g$; or
3. $t \geq_W s$, $h = g$, and either
   (a) $h \notin \Sigma_{AC}$ and $(t_1, \dots, t_n) \succ^{lex}_{\mathtt{ACKBO}} (s_1, \dots, s_n)$; or
   (b) $h \in \Sigma_{AC}$ and
      i. $\dot{\{} t_1, \dots, t_n \dot{\}} \succ_h \dot{\{} s_1, \dots, s_k \dot{\}}$; or
      ii. $\dot{\{} t_1, \dots, t_n \dot{\}} \succeq_h \dot{\{} s_1, \dots, s_k \dot{\}}$ and $n > k$; or
      iii. $\dot{\{} t_1, \dots, t_n \dot{\}} \succeq_h \dot{\{} s_1, \dots, s_k \dot{\}}$, $n = k$ and $\dot{\{} t_1, \dots, t_n \dot{\}} \succ^{mul}_{\mathtt{ACKBO}} \dot{\{} s_1, \dots, s_k \dot{\}}$. □

The proof that $\succ_{\mathtt{ACKBO}}$ is an AC-compatible simplification order is similar to the ground case, so we have the following theorem.

THEOREM 8.6.8 *The relation $\succ_{\mathtt{ACKBO}}$ is an AC-compatible monotonic order satisfying the subterm property. Moreover, on ground terms it coincides with the order of Definition 8.5.5.* □

THEOREM 8.6.9 *$\succ_{\mathtt{ACKBO}}$ is closed under substitutions, that is, if $t \succ_{\mathtt{ACKBO}} s$, then for every substitution $\theta$, $t\theta \succ_{\mathtt{ACKBO}} s\theta$.* □

## 8.7 Related Work

In general, Knuth-Bendix orders and recursive path orders are incomparable in the sense that there are rewrite (equational) systems that can be oriented by Knuth-Bendix orders but cannot be oriented by recursive path orders, and vice versa. To compare Knuth-Bendix orders with orders based on polynomial interpretations (or combinations of polynomial interpretations with recursive path orders) let us note that usually it is difficult to find a suitable polynomial interpretation which orients a given rewrite (equational) system. For Knuth-Bendix orders, we can employ efficient algorithms (see Chapters 6,7).

An attempt to define an AC-compatible Knuth-Bendix order was undertaken in [Steinbach 1990] for a special case when each AC-symbol $+$ is of the weight 0 and is also a maximal symbol w.r.t. $\gg$. It is proposed to compare terms with the top symbol $+$ first by weight and then by comparing the multisets of their arguments. Let us give an example demonstrating that the order defined in this way lacks the monotonicity property.

Consider the weight function $w$ such that $w(+) = 0$ and $w(c) = w(d) = w(g) = 1$ and a precedence relation $\gg$ such that $+ \gg g$. Let $t = c + d$ and $s = g(c)$. Then $|t| = |s|$, and therefore $t \succ_{\mathtt{ACKBO}} s$. Take any term $r$. Then by monotonicity we must have $r + c + d \succ_{\mathtt{ACKBO}} r + g(c)$. But in fact we have $r + g(c) \succ_{\mathtt{ACKBO}} r + c + d$, since $|g(c)| > |c|$ and $|g(c)| > |d|$.

For future research let us mention the problems of constraint solving and orientability for AC-compatible Knuth-Bendix orders. It is worth to note that algorithms and complexity results for constraint solving for AC-RPO are presented in [Comon, Nieuwenhuis and Rubio 1995, Godoy and Nieuwenhuis 2001].

Undecidability of first-order constraint solving in the presence of AC-symbols follows from the results in [Treinen 1990], where it is shown that already the $\Sigma_3$ fragment of the first-order theory of any trem algebra modulo associativity and commutativity is undecidable for signatures that contain at least one constant, one non-constant function symbol and one AC function symbol.

# Chapter 9

# Conclusions

In this thesis we have presented results of our study of decision problems for Knuth–Bendix orders that have applications in automated deduction. Let us summarise our main results and point out some related open problems.

**Constraint solving.** Ordering constraints are crucial for pruning search space in theorem provers. As a consequence algorithms for solving various ordering constraints are of great importance. In this thesis we have shown that the problem of solving Knuth-Bendix ordering constraints is decidable and NP–complete. We have presented an algorithm for solving Knuth–Bendix ordering constraints with an optimal complexity bound. Our algorithm extensively uses nondeterministic choices. It would be interesting to investigate how this nondeterminism can be reduced. Another problem to study is solving Knuth-Bendix ordering constraints under the extended signature semantics.

Constraints consisting of single inequalities are commonly used in automated theorem proving. We have presented a polynomial-time algorithm for solving Knuth-Bendix ordering constraints consisting of single inequalities. We believe that this algorithm can be efficiently implemented. It can also be used to approximate solving general Knuth-Bendix ordering constraints.

We have also been studying the constraint solving problem for first-order constraints. We have shown the decidability of the first-order Knuth-Bendix ordering constraints over unary signatures. Our decision procedure uses interpretation of unary terms as trees and uses decidability of the weak monadic second-order theory of binary trees. Although decision procedures for weak monadic second-order theory of binary trees behave reasonably well in practice, theoretical lower bound for complexity of this problem is nonelementary. An exact complexity for the problem of solving first-order Knuth-Bendix ordering constraints over unary signatures remains unknown. A more general open problem is the decidability and complexity of full first-order theory of Knuth-Bendix orders.

**Orientability.** Orientability by simplification orders is useful in term rewriting for showing termination of term rewriting systems, and in theorem proving for finding effective strategies for particular problems and theories. In automated systems which are dealing with equality it is desirable to have an efficient algorithm for orientability of systems consisting of term rewrite rules and equalities. We have shown that such an algorithm exists for Knuth-Bendix orders. In particular, we present a polynomial-time algorithm which checks for a given system of term rewriting rules and equalities whether there exists a Knuth-Bendix order which orients this system, and if such an order exists, the algorithm finds parameters of this order. To complete the complexity analysis of this problem we have shown that the orientability problem for Knuth-Bendix orders is P-complete even for systems consisting only of rewrite rules or only of equalities. A direction for future research is to integrate our orientability algorithm into existing theorem provers and assess its usefulness experimentally.

**AC-compatible Knuth-Bendix orders.** Axioms of associativity and commutativity occur in many important theories. Unfortunately these axioms are very difficult to deal with since they are extremely prolific due to non-orientability of the commutativity axiom. The main approach to overcome this problem is to integrate AC-reasoning into inference systems, which requires total AC-compatible simplification orders. The importance of AC-compatible orders triggered a huge amount of research devoted for designing such orders, mostly by modifying recursive path orders. We have shown that it is possible to modify Knuth-Bendix orders to AC-compatible orders. Moreover, these orders preserve attractive properties of original Knuth-Bendix orders such as a polynomial-time algorithm for term comparison and computationally efficient approximations based on weight comparisons. We believe that the algorithm for comparing terms in these orders can be efficiently implemented. For future research let us mention the problems of constraint solving and orientability for AC-compatible Knuth-Bendix orders.

# References

Arts T. and Giesl J. [2000], 'Termination of term rewriting using dependency pairs', *Theoretical Computer Science* **236**(1-2), 133–178.

Arts T. and Giesl J. [2001], 'Verification of erlang processes by dependency pairs', *Applicable Algebra in Engineering, Communication and Computing* **12**(1/2), 39–72.

Baader F. and Nipkow T. [1998], *Term Rewriting and and All That*, Cambridge University press, Cambridge.

Baaz M., Egly U. and Leitsch A. [2001], Normal form transformations, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 5, pp. 273–333.

Bachmair L. [1992], 'Associative-commutative reduction orderings', *Information Processing Letters* **43**(1), 21–27.

Bachmair L. and Ganzinger H. [2001], Resolution theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 2, pp. 19–99.

Bachmair L. and Plaisted D. [1985], Associative path orderings, *in* 'Rewriting Techniques and Applications', Vol. 202 of *Lecture Notes in Computer Science*, pp. 241–254.

Belegradek O. [1988], Model theory of locally free algebras (in Russian), *in* 'Model Theory and its Applications', Vol. 8 of *Trudy Instituta Matematiki*, Nauka, Novosibirsk, pp. 3–24. English translation in Translations of the American Mathematical Society.

Brand D. [1975], 'Proving theorems with the modification method', *SIAM Journal of Computing* **4**, 412–430.

Bürkert H.-J. [1990], A resolution principle for clauses with constraints, *in* M. Stickel, ed., 'Proc. 10th CADE', Vol. 449 of *Lecture Notes in Artificial Intelligence*, pp. 178–192.

Cherifa A. B. and Lescanne P. [1987], 'Termination of rewriting systems by polynomial interpretations and its implementation', *Science of Computer Programming* **9**(2), 137–159.

Comon H. [1990], 'Solving symbolic ordering constraints', *International Journal of Foundations of Computer Science* **1**(4), 387–411.

Comon H., Dauchet M., Gilleron R., Jacquemard F., Lugiez D., Tison S. and Tommasi M. [1997], 'Tree automata techniques and applications', Available on: http://www.grappa.univ-lille3.fr/tata.

Comon H. and Lescanne P. [1989], 'Equational problems and disunification', *Journal of Symbolic Computation* **7**(3,4), 371–425.

Comon H., Nieuwenhuis R. and Rubio A. [1995], Orderings, AC-theories and symbolic constraint solving, *in* 'Proc. 10th IEEE Symposium on Logic in Computer Science, (LICS'95)', IEEE Computer Society Press, pp. 375–385.

Comon H. and Treinen R. [1994], Ordering constraints on trees, *in* S. Tison, ed., 'Trees in Algebra and Programming: CAAP'94', Vol. 787 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 1–14.

Comon H. and Treinen R. [1997], 'The first-order theory of lexicographic path orderings is undecidable', *Theoretical Computer Science* **176**(1–2), 67–87.

Cormen T., Leiserson C. and Rivest R. [1991], *Introduction to Algorithms*, The MIT Press.

Dershowitz N. [1979], 'A note on simplification orderings', *Information Processing Letters* **9**(5), 212–215.

Dershowitz N. [1982], 'Orderings for term rewriting systems', *Theoretical Computer Science* **17**, 279–301.

Dershowitz N., Hsiang J., Josephson A. and Plaisted D. [1983], Associative-commutative rewriting., *in* 'Proc. International Joint Conference on Artificial Intelligence (IJCAI)', pp. 940–944.

Dershowitz N. and Manna Z. [1979], Proving termination with multiset orderings, *in* 'Proceedings of the 6th International Colloquium on Automata, Languages and Programming (ICALP'79)', Vol. 71 of *Lecture Notes in Computer Science*, European Association of Theoretical Computer Science, Springer-Verlag, Berlin, pp. 188–202.

Dick J., Kalmus J. and Martin U. [1990], 'Automating the Knuth-Bendix ordering', *Acta Informatica* **28**(2), 95–119.

Ershov Y. [1980], *Problems of decidability and constructive models*, Nauka. (in Russian).

Ferrante J. and Rackoff C. [1979], *The computational complexity of logical theories*, Vol. 718 of *Lecture Notes in Mathematics*, Springer-Verlag.

Gnaedig I. and Lescanne P. [1986], Proving termination of associative commutative rewriting systems by rewriting, *in* 'International Conference on Automated Deduction', Vol. 230 of *Lecture Notes in Computer Science*, pp. 52–61.

Godoy G. and Nieuwenhuis R. [2001], On ordering constraints for deduction with built-in abelian semigroups, monoids and groups, *in* 'Proc. 16th IEEE Symposium on Logic in Computer Science, (LICS'01)', IEEE Computer Society Press, pp. 38–50. journal version to appear.

Hillenbrand T., Buch A., Vogt R. and Löchner B. [1997], 'Waldmeister: High-performance equational deduction', *Journal of Automated Reasoning* **18**(2), 265–270.

Hodges W. [1993], *Model theory*, Cambridge University Press.

Hoe J. C. and Arvind [1999], Hardware synthesis from term rewriting systems, *in* '10th International Conference on Very Large Scale Integration (VLSI '99)', Vol. 162 of *IFIP Conference Proceedings*, pp. 595–619.

Hsiang J. and Rusinowitch M. [1986], A new method for establishing refutational completeness in theorem proving, *in* 'Proc. 8th CADE', Vol. 230 of *Lecture Notes in Computer Science*, pp. 141–152.

Huet G. [1972], Constrained Resolution: A Complete Method for Type Theory, PhD thesis, Computing and Information Sciences, Case Western Reserve University.

Jouannaud J.-P. and Okada M. [1991], Satisfiability of systems of ordinal notations with the subterm property is decidable, *in* 'Proc. of 18th International Colloquium on Automata, Languages and Programming (ICALP'91)', Vol. 510 of *Lecture Notes in Computer Science*, pp. 455–468.

Kamin S. and Lévy J.-J. [1980], Attempts for generalizing the recursive path ordering. Unpublished manuscript.

Kapur D. and Sivakumar G. [1997], A total, ground path ordering for proving termination of AC-rewrite systems, *in* 'Rewriting Techniques and Applications', Vol. 1232 of *Lecture Notes in Computer Science*, pp. 142–156.

Kapur D. and Sivakumar G. [1998], Proving associative-communicative termination using RPO-compatible orderings, *in* 'Automated Deduction in Classical and Non-Classical Logics, Selected Papers', Vol. 1761 of *Lecture Notes in Computer Science*, pp. 39–61.

KAPUR D., SIVAKUMAR G. AND ZHANG H. [1990], A new method for proving termination of AC-rewrite systems, *in* 'Proceedings of the 10th Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS'90)', Vol. 472 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 133–148.

KAPUR D., SIVAKUMAR G. AND ZHANG H. [1995], 'A path ordering for proving termination of AC rewrite systems', *Journal of Automated Reasoning* **14**(2), 293–316.

KIRCHNER C., KIRCHNER H. AND RUSINOWITCH M. [1990], 'Deduction with symbolic constraints', *Revue Francaise d'Intelligence Artificielle* **4**(3), 9–52. Special issue on automated deduction.

KNUTH D. AND BENDIX P. [1970], Simple word problems in universal algebras, *in* J. Leech, ed., 'Computational Problems in Abstract Algebra', Pergamon Press, Oxford, pp. 263–297.

KOROVIN K. AND VORONKOV A. [2000], A decision procedure for the existential theory of term algebras with the Knuth–Bendix ordering, *in* 'Proc. 15th Annual IEEE Symp. on Logic in Computer Science (LICS'00)', Santa Barbara, California, pp. 291–302.

KOROVIN K. AND VORONKOV A. [2001*a*], Knuth–Bendix constraint solving is NP-complete, *in* 'Proceedings of 28th International Colloquium on Automata, Languages and Programming (ICALP'01)', Vol. 2076 of *Lecture Notes in Computer Science*, Springer, pp. 979–992. journal version in [Korovin and Voronkov 2003*b*].

KOROVIN K. AND VORONKOV A. [2001*b*], Verifying orientability of rewrite rules using the Knuth–Bendix order, *in* 'Proc. 10th International Conference on Rewriting Techniques and Applications (RTA'01)', Vol. 2051 of *Lecture Notes in Computer Science*, Springer, pp. 137–153. journal version in [Korovin and Voronkov 2003*d*].

KOROVIN K. AND VORONKOV A. [2002], The decidability of the first-order theory of the Knuth-Bendix order in the case of unary signatures, *in* 'Proceedings of the 22th Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS'02)', Vol. 2556 of *Lecture Notes in Computer Science*, Springer, pp. 230–240.

KOROVIN K. AND VORONKOV A. [2003*a*], An AC-compatible Knuth–Bendix order, *in* '19th International Conference on Automated Deduction (CADE'03)', Vol. 2741 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 47 – 59.

KOROVIN K. AND VORONKOV A. [2003*b*], 'Knuth–Bendix constraint solving is NP-complete', *ACM Transactions on Computational Logic* . to appear.

Korovin K. and Voronkov A. [2003*c*], Orienting equalities with the Knuth-Bendix order, *in* 'Proc. 18th IEEE Symposium on Logic in Computer Science, (LICS'03)', IEEE Computer Society Press, pp. 75–84.

Korovin K. and Voronkov A. [2003*d*], 'Orienting rewrite rules with the Knuth–Bendix order', *Information and Computation* **183**(2), 165–186.

Krishnamoorthy M. and Narendran P. [1985], 'On recursive path ordering', *Theoretical Computer Science* **40**, 323–328.

Kunen K. [1987], 'Negation in logic programming', *Journal of Logic Programming* **4**, 289–308.

Lescanne P. [1984], Term rewriting systems and algebra, *in* R. Shostak, ed., '7th International Conference on Automated Deduction, CADE-7', Vol. 170 of *Lecture Notes in Computer Science*, pp. 166–174.

Maher M. [1988], Complete axiomatizations of the algebras of finite, rational and infinite trees, *in* 'Proc. IEEE Conference on Logic in Computer Science (LICS)', pp. 348–357.

Maĺcev A. [1961], 'On the elementary theories of locally free universal algebras', *Soviet Mathematical Doklady* **2**(3), 768–771.

Marché C. [1995], Associative-commutative reduction orderings via head-preserving interpretations, Technical Report 95–2, E.N.S. de Cachan.

Martin U. [1987], How to choose weights in the Knuth–Bendix ordering, *in* 'Rewriting Technics and Applications', Vol. 256 of *Lecture Notes in Computer Science*, pp. 42–53.

Martin U. and Shand D. [2000], 'Invariants, patterns and weights for ordering terms', *Journal of Symbolic Computation* **29**(6), 921–957.

Narendran P. and Rusinowitch M. [1991], Any ground associative-commutative theory has a finite canonical system, *in* 'Rewriting Techniques and Applications', Vol. 488 of *Lecture Notes in Computer Science*, pp. 423–434.

Narendran P. and Rusinowitch M. [2000], The theory of total unary RPO is decidable, *in* 'Proceedings of First International Conference on Computational Logic, CL'2000', Vol. 1861 of *Lecture Notes in Computer Science*, pp. 660–672.

Narendran P., Rusinowitch M. and Verma R. [1998], RPO constraint solving is in NP, *in* 'CSL: 12th Workshop on Computer Science Logic', Vol. 1584 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 385–398.

NIEUWENHUIS R. [1993], 'Simple LPO constraint solving methods', *Information Processing Letters* **47**, 65–69.

NIEUWENHUIS R. AND RIVERO J. [1999], Solved forms for path ordering constraints, *in* 'Proc. 10th International Conference on Rewriting Techniques and Applications (RTA)', Vol. 1631 of *Lecture Notes in Computer Science*, Springer Verlag, Trento, Italy, pp. 1–15. journal version in [Nieuwenhuis and Rivero 2002].

NIEUWENHUIS R. AND RIVERO J. [2002], 'Practical algorithms for deciding path ordering constraint satisfaction', *Information and Computation* **178**(2), 422–440.

NIEUWENHUIS R. AND RUBIO A. [1992], Theorem proving with ordering constrained clauses, *in* '11th International Conference on Automated Deduction (CADE'92)', Vol. 607 of *Lecture Notes in Computer Science*, pp. 477–491.

NIEUWENHUIS R. AND RUBIO A. [1995], 'Theorem proving with ordering and equality constrained clauses', *Journal of Symbolic Computation* **19**(4), 321–351.

NIEUWENHUIS R. AND RUBIO A. [1997], 'Paramodulation with built-in AC-theories and symbolic constraints', *Journal of Symbolic Computation* **23**(1), 1–21.

NIEUWENHUIS R. AND RUBIO A. [2001], Paramodulation-based theorem proving, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 7, pp. 371–443.

NONNENGART A. AND WEIDENBACH C. [2001], Computing small clause normal forms, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. I, Elsevier Science, chapter 6, pp. 335–367.

PAPADIMITRIOU C. [1994], *Computational Complexity*, Addison-Wesley.

PETERSON G. [1983], 'A technique for establishing completeness results in theorem proving with equality', *SIAM Journal of Computing* **12**(1), 82–100.

RABIN M. O. [1977], Decidable theories, *in* J. Barwise, ed., 'Handbook of Mathematical Logic', Vol. 90 of *Studies in Logic and the Foundations of Mathematics*, North-Holland, Amsterdam, chapter C.3, pp. 595–629.

REYNOLDS J. [1965], 'Unpublished seminar notes', Stanford University, Palo Alto.

RIAZANOV A. AND VORONKOV A. [1999], Vampire, *in* H. Ganzinger, ed., '16th International Conference on Automated Deduction (CADE'99)', Vol. 1632 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Trento, Italy, pp. 292–296.

ROBINSON G. AND WOS L. [1969], Paramodulation and theorem-proving in first order theories with equality, *in* B. Meltzer and D. Michie, eds, 'Machine Intelligence', Vol. 4, Edinburgh University Press, pp. 135–150.

ROBINSON J. [1965], 'A machine-oriented logic based on the resolution principle', *Journal of the Association for Computing Machinery* **12**(1), 23–41.

RUBIO A. [1999], A fully syntactic AC-RPO, *in* 'Rewriting Techniques and Applications', Vol. 1631 of *Lecture Notes in Computer Science*, pp. 133–147.

RUBIO A. [2002], 'A fully syntactic AC-RPO', *Information and Computation* **178**(2), 515–533.

RUBIO A. AND NIEUWENHUIS R. [1993], A precedence-based total AC-compatible ordering, *in* 'Rewriting Techniques and Applications', Vol. 690 of *Lecture Notes in Computer Science*, pp. 374–388. journal version in [Rubio and Nieuwenhuis 1995].

RUBIO A. AND NIEUWENHUIS R. [1995], 'A total AC-compatible ordering based on RPO', *Theoretical Computer Science* **142**(2), 209–227.

RUSINOWITCH M. AND VIGNERON L. [1995], 'Automated deduction with associative-commutative operators', *Applicable Algebra in Engineering, Communication and Computing* **6**(1), 23–56.

SCHRIJVER A. [1998], *Theory of Linear and Integer Programming*, John Wiley and Sons.

SCHULZ S. [1999], System abstract: E 0.3, *in* H. Ganzinger, ed., '16th International Conference on Automated Deduction (CADE'99)', Vol. 1632 of *Lecture Notes in Artificial Intelligence*, Springer Verlag, Trento, Italy, pp. 297–301.

SLAGLE J. R. [1967], 'Automatic theorem proving with renamable and sematic resolution', *Communications of the ACM* **14**, 687–697.

STEINBACH J. [1990], AC-termination of rewrite systems: A modified Knuth-Bendix ordering, *in* 'Algebraic and Logic Programming', Vol. 463 of *Lecture Notes in Computer Science*, pp. 372–386.

THATCHER J. W. AND WRIGHT J. B. [1968], 'Generalized finite automata theory with an application to a decision problem of second-order logic', *Mathematical Systems Theory* **2(1)**, 57–81.

TREINEN R. [1990], A new method for undecidability proofs of first order theories, *in* 'Proceedings of the 10th Conference on Foundations of Software Technology and Theoretical Computer Science, (FSTTCS'90)', Vol. 472 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 48–62. journal version in [Treinen 1992].

TREINEN R. [1992], 'A new method for undecidablity proofs of first order theories', *Journal of Symbolic Computation* **14**(5), 437–458.

Venkataraman K. [1987], 'Decidability of the purely existential fragment of the theory of term algebras', *Journal of the Association for Computing Machinery* **34**(2), 492–510.

Voronkov A. [2000], Formulae over reduction orderings: some solved and yet unsolved problems. invited talk RTA'00.

Weidenbach C. [2001], Combining superposition, sorts and splitting, *in* A. Robinson and A. Voronkov, eds, 'Handbook of Automated Reasoning', Vol. II, Elsevier Science, chapter 27, pp. 1965–2013.

# Index

### W