Knuth–Bendix Constraint Solving Is NP-Complete

KONSTANTIN KOROVIN
Max-Planck-Institut für Informatik, Saarbrücken, Germany
and
ANDREI VORONKOV
The University of Manchester, UK

We show the NP-completeness of the existential theory of term algebras with the Knuth–Bendix order by giving a nondeterministic polynomial-time algorithm for solving Knuth–Bendix ordering constraints.

Categories and Subject Descriptors: F.4.1 [**Mathematical Logic and Formal Languges**]: Mathematical Logic—*Computational logic*; I.2.3 [**Artificial Intelligence**]: Deduction and Theorem Proving—*Resolution*

General Terms: Theory

Additional Key Words and Phrases: Ordering constraints, automated deduction, Knuth–Bendix orders

1. INTRODUCTION

Solving ordering constraints in term algebras with various reduction orders is used in rewriting to prove termination of recursive definitions and in automated deduction to prune the search space [Comon 1990; Kirchner 1995; Nieuwenhuis 1999; Nieuwenhuis and Rubio 2001; Ganzinger and Nieuwenhuis 2001]. Nieuwenhuis [1999] connects further progress in automated deduction with constraint-based deduction.

Two kinds of orders are used in automated deduction: the Knuth–Bendix order [Knuth and Bendix 1970] and various versions of recursive path orders [Dershowitz 1982; Kamin and Lévy 1980]. The Knuth–Bendix order is

A. Voronkov was partially supported by a grant from EPSRC.
Authors' addresses: K. Korovin, Max-Planck-Institut für Informatik, Stuhlsatzenhausweg 85, D-66123, Saarbrücken, Germany; email: korovin@mpi-sb.mpg.de; A. Voronkov, Department of Computer Science, University of Manchester, Oxford Road, Manchester M13 9PL, UK; email: voronkov@cs.man.ac.uk.

used in the state-of-the-art theorem provers, for example, E [Schulz 2002], SPASS [Weidenbach et al. 1999], Vampire [Riazanov and Voronkov 2002], and Waldmeister [Löchner and Hillenbrand 2002]. There is extensive literature on solving recursive path ordering constraints (e.g., Comon [1990], Jouannaud and Okada [1991], Nieuwenhuis [1993], and Narendran et al. [1999]). The decidability of Knuth–Bendix ordering constraints was proved only recently in Korovin and Voronkov [2000]. The algorithm described in that paper shows that the problem belongs to 2-NEXPTIME. It was also shown that the problem is NP-hard by reduction of the solvability of systems of linear Diophantine equations to the solvability of Knuth–Bendix ordering constraints. In this aricle, 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 [Korovin and Voronkov 2001].

This article is structured as follows. In Section 2, we define the main notions of this paper. In Section 3, 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, we present such an algorithm using transformation to systems of linear Diophantine inequalities over the weights of variables. Finally, in Section 5, we complete the proof of the main result and present some examples. Section 6, discusses related work and open problems.

2. PRELIMINARIES

We call a *signature* a finite set of function symbols with associated arities. In this article we assume an arbitrary but fixed signature Σ. *Constants* are function symbols of the arity 0. We assume that Σ contains at least one constant. We denote variables by x, y, z and terms by r, s, t. The set of all ground terms of the signature Σ can be considered as the *term algebra* of this signature, $\mathrm{TA}(\Sigma)$, by defining the interpretation $g^{\mathrm{TA}(\Sigma)}$ of any function symbol g by $g^{\mathrm{TA}(\Sigma)}(t_1, \dots, t_n) = g(t_1, \dots, t_n)$. For details, see, for example, Hodges [1993] or Maher [1988]. It is easy to see that in term algebras any ground term is interpreted by itself.

Denote the set of natural numbers by $\mathbb{N}$. The Knuth–Bendix order is a family of orders parameterized by two parameters: a weight function and a precedence relation.

Definition 2.1 (*Weight Function*). We call a *weight function* on Σ any function $w : \Sigma \to \mathbb{N}$ such that (i) $w(a) > 0$ for every constant $a \in \Sigma$, (ii) there exists 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 $|g(t_1, \dots, t_n)| = w(g) + |t_1| + \dots + |t_n|$.

These conditions on the weight function ensure that the Knuth–Bendix order is a simplification order total on ground terms (see, e.g., Baader and Nipkow [1998]). In this article, *f will always denote a unary function symbol of the weight* 0.

The following lemma is straightforward.

LEMMA 2.2. *Every weight function satisfies the following properties.*

(1) *The weight of every ground term is positive.*
(2) *If Σ 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. If Σ contains the unary function symbol of the weight* 0*, then every weight contains either no terms at all or an infinite number of different terms.*
(3) *If a term s is a subterm of t and $|s| = |t|$, then t has the form $f^m(s)$ for some m (recall that f is the function symbol of the weight* 0*).*

Definition 2.3. A *precedence relation* on Σ is any total order $\gg$ on Σ. A precedence relation $\gg$ is said to be *compatible* with a weight function w if the existence of a unary function symbol f of the weight zero implies that f is the greatest element with respect to $\gg$.

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

Definition 2.4. The *Knuth–Bendix order* on TA(Σ) is the binary relation $\succ$ defined as follows: For any ground terms $t = g(t_1, \dots, t_n)$ and $s = h(s_1, \dots, s_k)$, we have $t \succ s$ 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 $1 \le i \le n$ we have $t_1 = s_1, \dots, t_{i-1} = s_{i-1}$ and $t_i \succ s_i$.

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 5.7 in Section 5).

The main result of this article is the following:

THEOREM 5.2. *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 5.1. *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 5.4. *For some Knuth-Bendix orders, the problem of solving ordering constraints is NP-complete.*

The proof of Theorem 5.2 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 $\succ$ 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 $\succ$ 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 Σ and natural numbers, respectively. The signature of $\mathrm{TA}^+(\Sigma)$ consists of

(1) all symbols of Σ interpreted as in $\mathrm{TA}(\Sigma)$;
(2) symbols $0, 1, >, +$ having their conventional interpretation over natural numbers;
(3) the binary relation symbol $\succ$ 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 2.5. *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 $\exists x_1 \cdots x_n A$, where $x_1, \dots, x_n$ are all variables of A, so the "only if" direction is trivial.

To prove the "if" direction, take any existential formula $\exists x_1, \dots, x_n A$. 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 $\neg s \succ t$ by $s =_{\mathrm{TA}} t \vee t \succ s$,
(2) any formula $\neg s =_{\mathrm{TA}} t$ by $s \succ t \vee t \succ s$,
(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 $\succ$ 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. □

Definition 2.6 (*Substitution, Solution*). A *substitution* is a mapping from the set of variables to the set of terms. A substitution θ 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 θ 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 θ is called a *solution* to a constraint C if θ 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 of solutions to A projected on $\bar{x}$.

3. 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 $C_1(\bar{x}, \bar{y}), \dots, C_n(\bar{x}, \bar{y})$ 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 $\mathrm{TA}^+(\Sigma)$:

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

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:

$$C(\bar{x}) \leftrightarrow \exists \bar{y} \exists \bar{z} (\mathit{f\text{-}variant}(\bar{x}, \bar{z}) \wedge (C_1(\bar{z}, \bar{y}) \vee \cdots \vee C_n(\bar{z}, \bar{y}))), \qquad (1)$$

where $f\text{-}\mathit{variant}(\bar{x}, \bar{z})$ 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) of the definition of the Knuth–Bendix order $|t| > |s|$ has a simple translation into a linear Diophantine inequality, but conditions (2) and (3) do not have one. 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 $|t| =_{\mathbb{N}} |s| \wedge t \succ s$. Obviously, $t_1 \succ t_2$ 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 $\succ$ is replaced by the new symbols $\succ_{lex}$ and $\succ_w$.

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

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 $t_1 \# t_2 \# \cdots \# t_n$, 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 3.1. *Any constraint C is equivalent to a disjunction $C_1 \vee \cdots \vee C_k$ 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

$$s \succ_w t \wedge C,\ s \succ_{lex} t \wedge C,\ s =_{\mathrm{TA}} t \wedge C,$$
$$t \succ_w s \wedge C,\ t \succ_{lex} s \wedge C.$$

By repeatedly doing this transformation we obtain a disjunction of constraints $C_1 \vee \cdots \vee C_k$ 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 $s_1\#s_2\# \cdots \#s_n\#s_1$, 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 $s_1\#s_2\# \cdots \#s_n \wedge s_1\#s_n$, $n \geq 2$. 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 $s_1\#s_2\# \cdots \#s_n$ is $\succ_w$, then $s_1 \succ_w s_n$ follows from $s_1\#s_2\# \cdots \#s_n$, otherwise $s_1\#s_2\# \cdots \#s_n$ 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 $s_1\#s_2\# \cdots \#s_n$ is $\succ_w$, then C_i is unsatisfiable. If all # in $s_1\#s_2\# \cdots \#s_n$ are $=_{\mathrm{TA}}$, then C_i is unsatisfiable too. Otherwise, all # in $s_1\#s_2\# \cdots \#s_n$ 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 $s_1\#s_2\# \cdots \#s_n$.
(3) *Case:* $s_1\#s_n$ *is* $s_1 =_{\mathrm{TA}} s_n$. If all # in $s_1\#s_2\# \cdots \#s_n$ are $=_{\mathrm{TA}}$, then $s_1 =_{\mathrm{TA}} s_n$ follows from $s_1\#s_2\# \cdots \#s_n$, 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. □

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

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

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

where $x_{11}, \dots, x_{kn_k}$ 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 3.1, 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, 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

$$C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}, \tag{2}$$

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 (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 (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 (2). Note that any chained constraint is of the form

$$\begin{array}{c} t_{11} \# t_{12} \# \cdots \# t_{1m_1} \\ \succ_w \\ \cdots \\ \succ_w \\ t_{k1} \# t_{k2} \# \cdots \# t_{km_k}, \end{array} \tag{3}$$

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 $t_{i1} \# t_{i2} \# \cdots \# t_{im_i}$ in which $\succ_w$ does not occur. So constraint (3) contains k rows, the first one is $t_{11} \# t_{12} \# \cdots \# t_{1m_1}$ and the last one $t_{k1} \# t_{k2} \# \cdots \# t_{km_k}$. Note that for any solution to (3) all terms in a row have the same weight.

LEMMA 3.2. *There exists a polynomial-time algorithm which transforms any chained constraint C into an equivalent chained constraint C' such that (i) the size of C' is not greater than the size of C; (ii) C' is either $\bot$ or of the form (3); and (iii) 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 θ 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 $y =_{\mathrm{TA}} \cdots =_{\mathrm{TA}} y$. 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 θ to C we have $|y\theta| = |t_{1n}\theta|$. Since t_{1n} is a flat term, by Lemma 2.2 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 3.2 and transform it into an equivalent disjunction of working constraints of smaller essential sizes in Lemma 3.5 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 (1) at the beginning of this section.

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 θ_1 and θ_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 θ_1 grounding for $\bar{x}$ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$, there exists a substitution θ_2 grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, and the restriction of θ_2 to $\bar{x}$ is an f-variant of θ_1.

(2) For every substitution θ_2 grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta_2$, there exists a substitution θ_1 such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta_1$ and θ_1 is an f-variant of the restriction of θ_2 to $\bar{x}$.

In other words, formula (1) 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 3.3. *Let $C = C_{chain} \wedge C_{arith} \wedge C_{triang} \wedge C_{simp}$ be a working constraint and θ_1 be a solution to C. Let θ_2 be an f-variant of θ_1 such that*

(1) *θ_2 is a solution to C_{chain} and*

(2) *θ_2 coincides with θ_1 on all variables not occurring in C_{chain}.*

Then there exists an f-variant θ_3 of θ_2 such that

(1) *θ_3 is a solution to C and*

(2) *θ_3 coincides with θ_2 on all variables except for the dependent variables of C_{triang}.*

PROOF. Let us first prove that θ_2 is a solution to both C_{arith} and C_{simp}. Since C_{simp} and C_{chain} have no common variables, it follows that θ_1 and θ_2 agree on all variables of C_{simp}, and so θ_2 is a solution to C_{simp}. Since θ_1 and θ_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, θ_2 is a solution to C_{arith} if and only if so is θ_1. So θ_2 is a solution to C_{arith}.

It is fairly easy to see that θ_2 can be changed on the dependent variables of C_{triang} obtaining a solution θ_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 Θ_1 of all solutions to C_{chain}, we can restrict ourselves to a subset Θ_2 of Θ_1 as soon as for every solution $\theta_1 \in \Theta_1$ there exists a solution $\theta_2 \in \Theta_2$ such that θ_2 is an f-variant of θ_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 that implies that any solution to C can be transformed into a solution with a "small" f-height.

LEMMA 3.4. *Let C_{chain} be a chained constraint of the form*

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

where each # is either $=_{\mathrm{TA}}$ or $\succ_{lex}$. Further, let C_{chain} satisfy the conditions of Lemma 3.2 and θ be a solution to C_{chain}. Then, there exists an f-variant θ' of θ such that

(1) *θ' 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) The row $p_l \# p_{l-1} \# \cdots \# p_1$ has a solution θ_1, such that (i) θ_1 is an f-variant of θ, (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) it is enough to show the following.

(5) There exists a solution θ_2 such that (i) θ_2 is an f-variant of θ, (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 (5), and hence (4). Since θ 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:

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

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 3.2, 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, (6) is proved.

Now note the following:

(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 (5), define the substitution θ_2 as follows:

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

By (6) and (7), θ_2 is defined correctly. We claim that θ_2 satisfies (5). The properties (i)–(iii) of (5) are straightforward by our construction, it only remains to prove that θ_2 is a solution to the row, that is, for every k' we have $p_{k'}\theta_2 \# p_{k'-1}\theta_2$. Consider the case when $k' > k$. Since θ 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 θ_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 θ is a solution to $p_k \succ_{lex} p_{k-1}$ and since θ_2 is an f-variant of θ, 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 (5), and hence of (4), is completed. In the same way as (4), we can also prove

(8) The constraint C_{chain} has a solution θ' such that (i) θ' is an f-variant of θ, (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, and (iv) θ' and θ coincide on all variables occurring in the rows below the first one.

It is easy to see that θ' from (8) satisfies all conditions required by our lemma. □

The following lemma is the main lemma of this section.

LEMMA 3.5. *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 that 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, that is, 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, that is, replace each of them by a 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 $p_l \# p_{l-1} \# \cdots \# p_1$. Then C_{chain} has the form $p_l \# p_{l-1} \# \cdots \# p_1 \succ_w t_1 \# \cdots \# t_n$. If $l = 1$, that is, 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 3.4, if some p_i is either a variable x or a term $f(x)$, it is enough to search for solutions θ 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 $x =_{\mathrm{TA}} t$ in S', such that x does not occur in f-terms of S'. We move such an equality $x =_{\mathrm{TA}} t$ 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 $0 \le r \le 1$. 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 $f^k(x) =_{\mathrm{TA}} f^m(y)$ do not occur in the first row.

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

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

(1) add $y_i =_{\mathrm{TA}} x_i$ to C_{triang};
(2) replace all occurrences of y_i in C_{chain} by x_i.

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

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

If all of the q_i are variables, we can move $q_n \succ_{lex} q_{n-1} \succ_{lex} \cdots \succ_{lex} q_1$ 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

$$\bigvee_{g \in \Sigma - \{f\}} \bigvee_{m=0\ldots l} x =_{\mathrm{TA}} f^m(g(\bar{y})), \tag{9}$$

where $\bar{y}$ is a sequence of pairwise different new variables. Since we proved that it is enough to restrict ourselves to solutions θ 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 $\perp$. 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

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

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

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

Then using the distributivity laws turn this formula into the equivalent disjunction of constraints of the form $C \wedge x_1 =_{\mathrm{TA}} y_1 \wedge \cdots \wedge x_{i-1} =_{\mathrm{TA}} y_{i-1} \wedge x_i \succ y_i$ for all $i = 1 \cdots u$. 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 $C_{chain} \wedge x_i \succ y_i$ into a disjunction of chained constraints as in Lemma 3.1.

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 $x_i \succ y_i$, 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 $f^k(h(u_1, \dots, u_m))$ or by $f^k(h(v_1, \dots, v_m))$ obtaining a constraint $f^k(h(u_1, \dots, u_m)) \succ_{lex} f^k(h(v_1, \dots, v_m))$, which is then replaced by

$$u_1 =_{\mathrm{TA}} v_1 \wedge \dots \wedge u_{i-1} =_{\mathrm{TA}} v_{i-1} \ \wedge u_i \succ v_i. \tag{10}$$

Let us call a variable u_i (respectively, v_i) *new* if $f^k(h(u_1, \dots, u_m))$ (respectively, $f^k(h(v_1, \dots, v_m))$) occurred in the terms of the first row when we replaced a variable by a nonvariable term containing h using formula (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 $x_i \succ x_j$, 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 $x_1 \# \cdots \# x_n \# x_1$ where $n \geq 2$ and at least one of the #'s is $\succ$. 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 parts. 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 $x =_{\mathrm{TA}} x_1 =_{\mathrm{TA}} \cdots =_{\mathrm{TA}} x_n$, 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 $\succ$. A new variable x is called *red in E* if it is not blue in E and E contains a chain $x \# x_1 \# \cdots \# x_n$, where x_n is an old variable, and all of the #'s are either $=_{\mathrm{TA}}$, or $\succ$, 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 (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 3.1, we can turn all the green variables into a disjunction of chained constraints of the form $v_1 \# \cdots \# v_n$, where # are $=_{\mathrm{TA}}$, $\succ_w$, or $\succ_{lex}$, and use the distributivity laws to obtain chained constraints $v_1 \# \cdots \# v_n$. 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 $v_1 \succ_{lex} \cdots \succ_{lex} v_k \succ_w v_{k+1} \# \cdots \# v_n$, we add $v_1 \succ_{lex} \cdots \succ_{lex} v_k$ to C_{simp} and $|v_k| > |v_{k+1}|$ to C_{arith}, and replace the green chain by $v_{k+1} \# \cdots \# v_n$. We do this transformation until the green chain becomes of the form $v_1 \succ_{lex} \cdots \succ_{lex} v_k$. 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 $f^k(h(u_1, \dots, u_m))$ in the first row at most one variable among $u_1, \dots, u_m$ 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 $u_1, \dots, u_{i-1}$ are blue. So suppose that u_i is either red or blue and $u_i \# y_n \# \cdots \# y_1$ is a shortest chain in E such that y_1 is old. We prove that the variables $u_1, \dots, u_{i-1}$ are blue, by induction on n. When $n = 1$ and u_i is red, E contains either $u_i \succ y_1$ or $y_1 \succ u_i$, where y_1 is old. Without loss of generality, assume that E contains $u_i \succ y_1$. Then (cf. (10)), this equation appeared in E when we replaced $f^k(h(u_1, \dots, u_m)) \succ_{lex} f^k(h(v_1, \dots, v_m))$ by $u_1 =_{\mathrm{TA}} v_1 \wedge \cdots \wedge u_{i-1} =_{\mathrm{TA}} v_{i-1} \wedge u_i \succ v_i$ and $y_1 = v_i$. But then E also contains the equations $u_1 =_{\mathrm{TA}} v_1, \dots, u_{i-1} =_{\mathrm{TA}} v_{i-1}$, where the variables $v_1, \dots, v_{i-1}$ are old, and so the variables $u_1, \dots, u_{i-1}$ are blue. In the same way, we can prove that, if u_i is blue, then $u_1, \dots, u_{i-1}$ are blue. The proof for $n > 1$ is similar, but we use the fact that $v_1, \dots, v_{i-1}$ 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 3.1.

Getting Rid of the Blue Variables. If E contains a blue variable x, then it also contains a chain of constraints $x =_{\mathrm{TA}} x_1 =_{\mathrm{TA}} \cdots =_{\mathrm{TA}} x_n$, where x_n is an old variable. We replace x by x_n in C and add $x =_{\mathrm{TA}} x_n$ 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. □

Lemmas 3.1 and 3.5 imply the following:

LEMMA 3.6. *Let C be a constraint. Then there exists a disjunction $C_1 \vee \cdots \vee C_n$ 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.*

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

4. 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 5, 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 $g_1, \dots, g_S$ of the signature Σ. We assume that the first B symbols $g_1, \dots, g_B$ is the sequence of all symbols in Σ of arity ≥ 2, and the first F symbols $g_1, \dots, g_F$ is the sequence of all nonconstant symbols in Σ. 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 $(n_1, \dots, n_S)$ 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 Σ 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 4.1. *The following relation $exists(x, n_1, \dots, n_S)$ is $\exists$-definable: there exists at least one ground term of Σ of the weight x and contents $(n_1, \dots, n_S)$.*

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

The first equation is

$$x = \sum_{1 \leq i \leq S} w(g_i) \cdot n_i. \tag{11}$$

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

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

$$1 + \sum_{1 \le i \le S} (arity_i - 1) \cdot n_i = 0. \tag{12}$$

□

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 ≥ 2, then there exists at least N different ground terms of the weight x.

LEMMA 4.2. *Let $x, n_1, \dots, n_S$ be natural numbers such that exists$(x, n_1, \dots, n_S)$ holds, $n_1 + \cdots + n_B \geq 1$ and $n_1 + \cdots + n_F \geq N$. Then, there exist at least N different ground terms with the contents $(n_1, \dots, n_S)$.*

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

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 than or equal to the number of constants in the signature.

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

LEMMA 4.4. *Let Σ 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 Σ contains a unary function symbol of the weight 0 then the number of different terms of any weight is either 0 or ω and the lemma trivially holds.

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

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

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, that is, there exists a term t of the weight x such that the number of occurrences of nonconstant function symbols in t is $M \le N$. Let $(n_1, \dots, n_S)$ be the contents of t and L denote the number of occurrences of constants in t. Note that (12) implies $L = 1 + \sum_{1 \le i \le F}(arity_i - 1) \cdot n_i$. Then, using (11), we obtain

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

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 4.2, 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, 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 $H \ge 1$, we have $M_3 > 0$. Let us show that $M_2 > 0$, i.e. $k > w(h) + w(c) \cdot (H - 1)$. We have

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

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. (11) and (12))

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

These 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 4.5. 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 4.6. *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 4.7. *Let the signature Σ 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, that is, 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 that, 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 Eqs. (11) and (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

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

If T_{i+1} has N or more terms, then, by Lemma 4.6, 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 $T_1 \cup \cdots \cup T_{i+1}$.

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 4.6.

(2) *At the end of step $i+1$ the set $T_1 \cup \cdots \cup T_{i+1}$ 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 $M + 1$. □

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

LEMMA 4.8. *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 Σ contains a unary function symbol of the weight, 0, then the number of different terms in any weight is either 0 or ω. Therefore, we can define $at\_least_N(x)$ as $\exists n_1 \cdots \exists n_S exists(x, n_1, \ldots, n_S)$.

Let us consider the case when the signature Σ consists of a unary function symbol g of a positive weight and constants. For every constant c in Σ, 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 Σ (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 4.4, 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 4.4 and 4.7, we have

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

5. MAIN RESULTS

In this section, we complete the proofs of the main results of this article.

THEOREM 5.1. *For every Knuth–Bendix order, the problem of solving ordering constraints is contained in NP.*

PROOF. Take a constraint. By Lemma 3.5, 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

$$C_{arith} \wedge C_{triang} \wedge C_{simp}. \tag{13}$$

Let C_{simp} contain a chain $x_1 \succ_{lex} \cdots \succ_{lex} x_N$ 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

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

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:

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

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 article. Let us call a signature Σ *trivial* if it consists of one constant symbol. Evidently, the first-order theory of the term algebra of a trivial signature is polynomial.

THEOREM 5.2. *The existential first-order theory of any term algebra of a nontrivial signature with the Knuth–Bendix order is NP-complete.*

PROOF. The containment in NP follows from Theorem 5.1. It is easy to prove NP-hardness by reducing propositional satisfiability to the existential theory of the algebra (even without the order). $\square$

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

Example 5.3. 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 Σ 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

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

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

$$x_1 + \cdots + x_n + k = x_0, \tag{14}$$

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

$$\begin{array}{l} \mathit{equal\_weight}(s^{k+2}(h(y_1, h(y_2, \ldots, \\ \qquad\qquad h(y_{n-1}, y_n)))), \\ \qquad s^{2n}(y_0)). \end{array} \tag{15}$$

It is not hard to argue that

(16) Formula (15) holds if and only if

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

Using (16), we can transform any system $D(x_0, \ldots, x_n)$ of linear Diophantine equations of the form (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

$$\begin{array}{l} \mathit{greater\_weight}(x, y) \leftrightarrow \\ \qquad s(x) \succ g(y) \end{array}$$

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

Since it is well-known that solving linear Diophantine equations is NP-hard, we have the following theorem.

THEOREM 5.4. *For some Knuth–Bendix orders, the problem of solving ordering constraints is NP-complete.*

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

LEMMA 5.5. *There exists a polynomial time algorithm that 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 $\Sigma = \{c_1, \ldots, c_n\}$, without loss of generality 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 with respect to $\succ_C$. There are two possible cases:

(1) x is a minimal term with respect to $\succ_C$, then we replace all occurrences of x in C by c_1.
(2) there exist some constants less than x with respect to $\succ_C$, then let c_{max} be the greatest with respect to $\succ$ constant among such constants. If c_{max} is the maximal constant in $\mathrm{TA}(\Sigma)$, 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) and (2) above, preserve satisfiability of constraints without equality. To this end, we consider a constraint C without equality and a solution θ to C. If the transformation (1) is applicable to C then it is easy to see that

$$\theta'(x) = \begin{cases} c_1, \text{ if } x \text{ is a minimal term with respect to } \succ_C, \\ \theta(x) \text{ otherwise.} \end{cases}$$

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

Similarly, one can show that the transformation (2) preserves satisfiability of constraints without equality. □

COROLLARY 5.6. *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 2, if we consider real-valued Knuth–Bendix orders then even comparison of ground terms might be undecidable. Let us show it by the following example.

Example 5.7. Consider a non-computable real number r such that $0 < r < 1$, that is, there is no algorithm that is, 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 $\succ$ 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 $g^m(c) \succ h^n(c) \succ g^{m-1}(c)$. 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 $m/n > r > (m - 1)/n$. 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.

6. RELATED WORK AND OPEN PROBLEMS

In this section, we overview previous work on Knuth–Bendix orders, recursive path orders, and extensions of term algebras with various relations.

The Knuth–Bendix order was introduced in Knuth and Bendix [1970]. Later, Dershowitz [1982] introduced recursive path orders (RPOs) and Kamin and Lévy [1980] lexicographic path orders (LPOs). A number of results on recursive path orders and solving LPO and RPO ordering constraints are known.

However, except for the very general result of Nieuwenhuis [1993], the techniques used for RPO constraints are not directly applicable to Knuth–Bendix orders. We used systems of linear Diophantine inequalities in our decidability proofs. This is not coincidental: Example 5.3 shows that systems of linear Diophantine inequalities are definable in the Knuth–Bendix order.

Comon and Treinen [1994] proved that LPO constraint solving is NP-hard already for constraints consisting of a single inequality. In Korovin and Voronkov [2001] we prove that the problem of solving Knuth–Bendix ordering constraints consisting of a single inequality can be solved in polynomial time.

In Korovin and Voronkov [2001], we present a polynomial time algorithm for the orientability problem: given a system of rewrite rules R, does there exist a Knuth–Bendix order which orients every ground instance of every rewrite rule in R. A similar problem of orientability for the non-ground version of the real-valued Knuth–Bendix order was studied by Dick, Kalmus, and Martin [Martin 1987; Dick et al. 1990] and an algorithm for orientability was given. Algorithms for, and complexity of, orientability problem for various versions of the recursive path orders were considered in Lescanne [1984], Detlefs and Forgaard [1985], and Krishnamoorthy and Narendran [1985]. In particular, in Krishnamoorthy and Narendran [1985], it is shown that the orientability problem by the non-ground version of the recursive path order is NP-complete. In automated deduction we often need to orient systems consisting of equations and term rewriting rules. In Korovin and Voronkov [2003a] we show that the orientability problem for systems of equations and term rewriting rules can be also solved in polynomial time for the Knuth–Bendix order.

Comon [1990] proved the decidability and Nieuwenhuis [1993] NP-completeness of LPO constraint solving. Jouannaud and Okada [1991] proved the decidability and Narendran et al. [1999] NP-completeness of RPO constraint solving. Recently, Nieuwenhuis and Rivero [1999] proposed a new efficient method for solving RPO constraints.

Lepper [2001] studies derivation length and order types of Knuth–Bendix orders, both for integer-valued and real-valued weight functions.

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].

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]. Term algebras with lexicographic path orders have an undecidable first-order theory [Comon and Treinen 1997]. However, if we consider term algebras over signatures consisting of unary symbols and constants then the first-order theory of lexicographic path orders over such term algebras is decidable [Narendran and Rusinowitch 2000]. In Korovin and Voronkov [2002], we show that the first-order theory of any Knuth–Bendix order over any term algebra over a signature consisting of unary function symbols and constants is decidable.

To conclude, we mention two open problems related to Knuth–Bendix orders. One problem is whether the whole first-order theory of Knuth–Bendix orders is decidable. Another problem is to describe the complexity of the constraint solving problem for Knuth–Bendix orders in the case of signatures consisting of unary function symbols and constants.

ACKNOWLEDGMENTS

We are grateful to reviewers of this paper for their detailed comments.

REFERENCES

BAADER, F. AND NIPKOW, T. 1998. *Term Rewriting and All That*. Cambridge University Press, Cambridge, mass.

BELEGRADEK, O. 1988. Model theory of locally free algebras (in Russian). In *Model Theory and its Applications*. Trudy Instituta Matematiki, vol. 8. Nauka, Novosibirsk, 3–24. (English translation in *Translations of the American Mathematical Society*.)

COMON, H. 1990. Solving symbolic ordering constraints. *Int. J. Found. Comput. Sci. 1*, 4, 387–411.

COMON, H. AND LESCANNE, P. 1989. Equational problems and disunification. *Journal of Symbolic Computations 7*, 3, 4, 371–425.

COMON, H. AND TREINEN, R. 1994. Ordering constraints on trees. In *Trees in Algebra and Programming: CAAP'94*, S. Tison, Ed. Lecture Notes in Computer Science, vol. 787. Springer-Verlag, New York, 1–14.

COMON, H. AND TREINEN, R. 1997. The first-order theory of lexicographic path orderings is undecidable. *Theoret. Comput. Sci. 176*, 1–2, 67–87.

DERSHOWITZ, N. 1982. Orderings for term rewriting systems. *Theoret. Comput. Sci. 17*, 279–301.

DETLEFS, D. AND FORGAARD, R. 1985. A procedure for automatically proving the termination of a set of rewrite rules. In *Rewriting Techniques and Applications, First International Conference, RTA-85* (Dijon, France), J.-P. Jouannaud, Ed. Lecture Notes in Computer Science, vol. 202. Springer-Verlag, New York, 255–270.

DICK, J., KALMUS, J., AND MARTIN, U. 1990. Automating the Knuth–Bendix ordering. *Acta Inf. 28*, 2, 95–119.

GANZINGER, H. AND NIEUWENHUIS, R. 2001. Constraints and theorem proving. In *Constraints in Computational Logics, International Summer School, September 5–8, 1999, Gif-sur-Yvette, France, Edition*, H. Comon, C. Marché, and R. Treinen, Eds. Lecture Notes in Computer Science, vol. 2002. Springer-Verlag, New York, 159–201.

HODGES, W. 1993. *Model theory*. Cambridge University Press, Cambridge, Mass.

JOUANNAUD, J.-P. AND OKADA, M. 1991. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Automata, Languages and Programming, 18th International Colloquium, ICALP'91* (Madrid, Spain), J. Albert, B. Monien, and M. Rodríguez-Artalejo, Eds. Lecture Notes in Computer Science, vol. 510. Springer-Verlag, New York, 455–468.

KAMIN, S. AND LÉVY, J.-J. 1980. Attempts for generalizing the recursive path orderings. Unpublished manuscript. (Available on the Web page of Pierre Lescanne: http.//perso.ens-lyon.fr/pierre.lescanne/not accessible.html.)

KIRCHNER, H. 1995. On the use of constraints in automated deduction. In *Constraint Programming: Basics and Tools*, A. Podelski, Ed. Lecture Notes in Computer Science, vol. 910. Springer-Verlag, New York, 128–146.

KNUTH, D. AND BENDIX, P. 1970. Simple word problems in universal algebras. In *Computational Problems in Abstract Algebra*, J. Leech, Ed. Pergamon Press, Oxford, 263–297.

KOROVIN, K. AND VORONKOV, A. 2000. A decision procedure for the existential theory of term algebras with the Knuth–Bendix ordering. In *Proceedings of the 15th Annual IEEE Symposium on Logic in Computer Science, (LICS'00)*. IEEE Computer Society Press, Los Alamitos, California, 291–302.

KOROVIN, K. AND VORONKOV, A. 2001. Verifying orientability of rewrite rules using the Knuth–Bendix order. In *Rewriting Techniques and Applications, 12th International Conference, RTA 2001*, A. Middeldorp, Ed. Lecture Notes in Computer Science, vol. 2051. Springer-Verlag, New York, 137–153. (Journal version: [Korovin and Voronkov 2003b].)

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)*. Lecture Notes in Computer Science, vol. 2556. Springer-Verlag, New York, 230–240.

KOROVIN, K. AND VORONKOV, A. 2003a. Orienting equalities with the Knuth–Bendix order. In *Proceedings of the 18th IEEE Symposium on Logic in Computer Science, (LICS'03)*. IEEE Computer Society Press, Los Alamitos, Calif., 75–84.

KOROVIN, K. AND VORONKOV, A. 2003b. Orienting rewrite rules with the Knuth–Bendix order. *Inf. Comput. 183*, 2, 165–186.

KRISHNAMOORTHY, M. AND NARENDRAN, P. 1985. On recursive path ordering. *Theoret. Comput. Sci. 40*, 323–328.

KUNEN, K. 1987. Negation in logic programming. *J. Logic Prog. 4*, 289–308.

LEPPER, I. 2001. Derivations lengths and order types of Knuth–Bendix orders. *Theoret. Comput. Sci. 269*, 1–2, 433–450.

LESCANNE, P. 1984. Term rewriting systems and algebra. In *Proceedings of the 7th International Conference on Automated Deduction, CADE-7*, R. Shostak, Ed. Lecture Notes in Computer Science, vol. 170. Springer Verlag, New York, 166–174.

LÖCHNER, B. AND HILLENBRAND, T. 2002. A phytography of WALDMEISTER. *AI Commun. 15*, 2–3, 127–133.

MAHER, M. 1988. Complete axiomatizations of the algebras of finite, rational and infinite trees. In *Proceedings of the IEEE Conference on Logic in Computer Science (LICS)*. IEEE Computer Society Press, Los Alamitos, Calif., 348–357.

MAĹCEV, A. 1961. On the elementary theories of locally free universal algebras. *Soviet Math. Dokl. 2*, 3, 768–771.

MARTIN, U. 1987. How to choose weights in the Knuth–Bendix ordering. In *Rewriting Technics and Applications*. Lecture Notes in Computer Science, vol. 256. Springer-Verlag, New York, 42–53.

NARENDRAN, P. AND RUSINOWITCH, M. 2000. The theory of total unary RPO is decidable. In *Proceedings of 1st International Conference on Computational Logic, CL'2000*. Lecture Notes in Computer Science, vol. 1861. Springer-Verlag, New York, 660–672.

NARENDRAN, P., RUSINOWITCH, M., AND VERMA, R. 1999. RPO constraint solving is in NP. In *Computer Science Logic, 12th International Workshop, CSL'98*, G. Gottlob, E. Grandjean, and K. Seyr, Eds. Lecture Notes in Computer Science, vol. 1584. Springer-Verlag, New York, 385–398.

NIEUWENHUIS, R. 1993. Simple LPO constraint solving methods. *Inf. Proc. Lett. 47*, 65–69.

NIEUWENHUIS, R. 1999. Rewrite-based deduction and symbolic constraints. In *Automated Deduction—CADE-16, 16th International Conference on Automated Deduction* (Trento, Italy). H. Ganzinger, Ed. Lecture Notes in Artificial Intelligence, vol. 1632. Springer-Verlag, New York, 302–313.

NIEUWENHUIS, R. AND RIVERO, J. 1999. Solved forms for path ordering constraints. In *Proceedings of the 10th International Conference on Rewriting Techniques and Applications (RTA)* (Trento, Italy). Lecture Notes in Computer Science, vol. 1631. Springer-Verlag, New York, 1–15.

NIEUWENHUIS, R. AND RUBIO, A. 2001. Paramodulation–based theorem proving. In *Handbook of Automated Reasoning*, A. Robinson and A. Voronkov, Eds. Vol. I. Elsevier Science, Amsterdam, The Netherlands, Chap. 7, 371–443.

RIAZANOV, A. AND VORONKOV, A. 2002. The design and implementation of Vampire. *AI Commun. 15*, 2–3, 91–110.

SCHULZ, S. 2002. E—A brainiac theorem prover. *AI Commun. 15*, 2–3, 111–126.

VENKATARAMAN, K. N. 1987. Decidability of the purely existential fragment of the theory of term algebras. *J. ACM 34*, 2, 492–510.

WEIDENBACH, C., AFSHORDEL, B., BRAHM, U., COHRS, C., ENGEL, T., KEEN, E., THEOBALT, C., AND TOPIC, D. 1999. System description: SPASS version 1.0.0. In *Automated Deduction—CADE-16, 16th International Conference on Automated Deduction* (Trento, Italy). H. Ganzinger, Ed. Lecture Notes in Artificial Intelligence, vol. 1632. Springer-Verlag, New York, 378–382.

Received September 2002; revised October 2003; accepted October 2003