A decision procedure for the existential theory of term algebras with the Knuth-Bendix ordering

Konstantin Korovin* Andrei Voronkov†

Department of Computer Science
University of Manchester
{korovin|voronkov}@cs.man.ac.uk

Abstract

We show the decidability of the existential theory of term algebras with any Knuth-Bendix ordering by giving a procedure for solving Knuth-Bendix ordering constraints.

1 Introduction

Solving ordering constraints in term algebras with various reduction orderings is used in automated deduction [**?**, Kirchner 1995, Nieuwenhuis 1999]. Nieuwenhuis [1999] connects further progress in automated deduction with constraint-based deduction.

Two kinds of orderings are used in automated deduction: Knuth-Bendix ordering [Knuth and Bendix 1970] and various versions of recursive path orderings [Dershowitz 1982]. Knuth-Bendix orderings are used in the state-of-the-art theorem provers, for example Vampire [Ryazanov and Voronkov 1999] and SPASS [Weidenbach 1999] (the winners of the last CASC competition [Sutcliffe 2000] in the mixed and first-order categories, respectively). There exists extensive literature on solving recursive path ordering constraints [e.g. **?**, Nieuwenhuis 1993], but no algorithms for solving Knuth-Bendix ordering constraints are known.

In this paper we prove that the problem of solvability of Knuth-Bendix ordering constraints is decidable. In fact, we prove a slight generalization of this result: the existential first-order theory of any term algebra with a Knuth-Bendix ordering is decidable.

As for complexity, NP-hardness of the set of satisfiable quantifier-free formulas can be shown in the same way as in [Nieuwenhuis 1993]. The algorithm presented here does not give an NP upper bound, we point out parts of our algorithm that may cause nonpolynomial behavior.

*Partially supported by grants from EPSRC and the Faculty of Science and Technology.

†Partially supported by grants from EPSRC and the Faculty of Science and Engineering.

This paper is structured as follows. In Section 2 we define all the main notions of this paper, including that of constraint, and formulate the main result. In Section 3 we introduce the notion of solved form of constraints and show that every constraint can be effectively transformed into an equivalent disjunction of solved forms. To decide solved forms, one has to be able to encode statements of the form: given a fixed number N and a variable number x, does there exists at least N terms of the weight x. In Section 4 we show how to encode such statements using linear Diophantine equations. Finally, in Section 5 we complete the proof of the main result. Section 6 discusses related word and open problems.

2 Preliminaries

We assume knowledge of *term algebras* (aka absolutely free algebras), the definition can be found in standard textbooks, e.g., [Hodges 1993]. We consider term algebras in a finite signature Σ with at least one constant, denoted $\mathrm{TA}(\Sigma)$. When it is convenient for us, we will regard constants as function symbols of arity 0. The elements of $\mathrm{TA}(\Sigma)$ are the ground terms of Σ, and every ground term t of the language is interpreted by t.

Let us now define Knuth-Bendix orderings on $\mathrm{TA}(\Sigma)$ [Knuth and Bendix 1970]. Denote the set of natural numbers by $\mathbb{N}$. The definition of Knuth-Bendix ordering is parametrized by a *weight function* on Σ, i.e., a function $w : \Sigma \to \mathbb{N}$, and a linear ordering $\gg$ on Σ. We require of a weight function the following: if $w(f) = 0$ and f is unary, then f must be the greatest w.r.t. $\gg$ in Σ, and weights of constants are positive. These conditions on the weight function ensure that the Knuth-Bendix ordering is a simplification ordering [see e.g. Baader and Nipkow 1998] total on ground terms. In this paper, f will always denote a unary function symbol of weight 0.

Given a weight function w on Σ, we define the *weight* $|t|$ of any ground term t as follows: $|c| = w(c)$ for any constant c and $|g(t_1, \ldots, t_n)| = w(g) + \sum_{i=1}^{n} |t_i|$.

Given a weight function w and a linear ordering $\gg$ on Σ, the *Knuth-Bendix ordering* on $\mathrm{TA}(\Sigma)$ is the binary relation $>_{KB}$ defined as follows. For any ground terms $g(t_1, \ldots, t_n)$ and $h(s_1, \ldots, s_k)$ we have $g(t_1, \ldots, t_n) >_{KB} h(s_1, \ldots, s_k)$ if

1. $|g(t_1, \ldots, t_n)| > |h(s_1, \ldots, s_k)|$
   or
2. $|g(t_1, \ldots, t_n)| = |h(s_1, \ldots, s_k)|$ and one of the following holds:
   (a) $g \gg h$ or
   (b) $g = h$ and for some $1 \leq i \leq n$ we have $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i >_{KB} s_i$.

Some authors [Martin 1987, Baader and Nipkow 1998] define a modification of Knuth-Bendix orderings in which weight function are real-valued. Knuth-Bendix orderings with real-valued weight functions are still reduction orderings. We do not consider real-valued weight functions in this paper, because for such functions even comparison of ground terms can be undecidable.

The main result of this paper is the following.

THEOREM 2.1 *The existential first-order theory of any term algebra with Knuth-Bendix ordering is decidable.*

The proof will be given after a series of lemmas. The idea of the proof is as follows. First, we will extend the language by the function defining the weight of terms and natural numbers with addition. Given an existential formula of the first-order theory of any term algebra with a Knuth-Bendix ordering, we will rewrite it step by step into an equivalent disjunction of existential formulas of the extended language. The main aim of these rewrite steps is to replace all occurrences of $>_{KB}$ by linear Diophantine equations on the weight of variables. At the end we will obtain existential formulas consisting of linear Diophantine equations on the weight of variables plus statements expressing that, for some fixed natural numbers N, there exists at least N terms of weight k. We will then prove that this statements can also be expressed using linear Diophantine equations and use the decidability of systems of linear Diophantine equations.

Our proof will not give precise upper bounds on complexity, we will only briefly address complexity-related questions.

In the sequel we assume a fixed signature Σ with a fixed weight function and ordering $\gg$ on Σ. 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 (i) of all symbols of Σ interpreted as in $\mathrm{TA}(\Sigma)$, (ii) symbols $0, 1, >, +$ having their conventional interpretation over natural numbers, (iii) the relation symbol $>_{KB}$, interpreted as the Knuth-Bendix ordering, (iv) the function symbol $|\ldots|$, interpreted as the weight function. When we need to distinguish the equality $=$ over the term algebra sort from the equality over the arithmetical sort, we denote the former by $=_{\mathrm{TA}}$, and the latter by $=_{\mathbb{N}}$.

We will prove decidability of the existential theory of $\mathrm{TA}^+(\Sigma)$, from which decidability of term algebras with a Knuth-Bendix ordering follows immediately.

We consider *satisfiability* 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. First, let us note that decidability of the existential theory of $\mathrm{TA}^+(\Sigma)$ is equivalent to decidability of constraint satisfaction.

PROPOSITION 2.2 *The existential theory of* $\mathrm{TA}^+(\Sigma)$ *is decidable if and only if so is the constraint satisfiability problem.*

PROOF. Obviously any instance A of constraint satisfiability problem can be considered as satisfiability 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 (i) any formula $\neg s >_{KB} t$ by $s =_{\mathrm{TA}} t \vee t >_{KB} s$, (ii) any formula $\neg s =_{\mathrm{TA}} t$ by $s >_{KB} t \vee t >_{KB} s$, (iii) any formula $\neg s > t$ by $s = t \vee t > s$; (iv) any formula $\neg s =_{\mathbb{N}} t$ by $s > t \vee t > s$, and convert A into disjunctive normal form. It is easy to see that we obtain a disjunction of constraints. The transformation gives an equivalent formula since both orderings $>_{KB}$ and $>$ are total. □

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 terms, the solutions $\bar{x}$ to C in $\mathrm{TA}^+(\Sigma)$ are exactly the solutions $\bar{x}, \bar{y}$ to A in $\mathrm{TA}^+(\Sigma)$ projected to $\bar{x}$.

3 Solved 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 weight 0, the transformation will preserve equivalence in the following 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 all $\bar{t}$, $C(\bar{t})$ holds if and only if there exists k and $\bar{s}$ such that $C_k(\bar{t},\bar{s})$. When the signature contains a unary function symbol of weight 0, the transformation will preserve a weaker form of equivalence: some solutions will be lost, but solvability will be preserved.

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

Let us split Knuth-Bendix ordering into two parts forced by the arithmetical condition on the weight of terms. Let $t_1 >_w t_2$ denote the formula $|t_1| > |t_2|$ and $t_1 \succ t_2$ denote the formula $t_1 >_{KB} t_2 \wedge |t_1| = |t_2|$. Obviously, $t_1 >_{KB} t_2$ if and only if $t_1 \succ t_2 \vee t_1 >_w t_2$. So without loss generality we can replace $>_{KB}$ with the new symbols $\succ$ and $>_w$.

We denote by $Arith$ any constraint using only arithmetical relations $=_{\mathbb{N}}$ and $>$ and function symbols, for example $|f(x)| > |a| + 3$. 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 *refined* if (1) it has a form $t_1\#t_2\#\ldots\#t_n$, where each $\#$ is $>_w$, $\succ$ or $=_{\mathrm{TA}}$, and (ii) each term t_i is flat.

LEMMA 3.1 *Any constraint C is equivalent to a disjunction of constraints of the form*

$$s_1\#s_2\#\ldots\#s_n \wedge Arith, \tag{1}$$

where $s_1\#s_2\#\ldots\#s_n$ is refined.

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 terms all terms will become flat.

Therefore, we assume that all terms occurring in C are flat. Let s,t be terms occurring in C. Using the valid formula $s >_w t \vee s \succ t \vee s =_{\mathrm{TA}} t \vee t >_w s \vee t \succ s$ we can replace C by disjunction of the constraints

$$\begin{array}{lll} s >_w t \wedge C, & s \succ t \wedge C, & s =_{\mathrm{TA}} t \wedge C, \\ t >_w s \wedge C, & t \succ s \wedge C. & \end{array}$$

By repeatedly doing this transformation with all pairs s,t occurring in C we obtain a disjunction of constraints of the required form. □

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

We say a constraint $y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_n =_{\mathrm{TA}} t_n$ is a *triangle form* if y_i does not occur in t_j for $j \geq i$. Denote by $\perp$ the logical constant "false". A constraint is in a *solved form* if it is either $\perp$ or has the form

$$\begin{array}{l} x_1\#x_2\#\ldots\#x_n \bigwedge \\ y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_k =_{\mathrm{TA}} t_k \bigwedge Arith, \end{array}$$

where

1. each occurrence of $\#$ is either $>_w$ or $\succ$;
2. $x_1,\ldots,x_n,y_1,\ldots,y_k$ are pairwise different variables;
3. the constraint $y_1 =_{\mathrm{TA}} t_1 \wedge \ldots \wedge y_k =_{\mathrm{TA}} t_k$ is in triangle form.

Our next aim is to transform any constraint C into an equally solvable disjunction of solved forms. By Lemma 3.1 we can assume that C is a conjunction of a refined constraint and arithmetical constraint of the form (1). To make it into a disjunction of solved form, we have to show how to replace the terms s_i by variables and eliminate $=_{\mathrm{TA}}$ among $\#$.

Note that any refined constraint is of the form

$$\begin{array}{c} t_{11}\#t_{12}\#\ldots\#t_{1m_1} \\ >_w \\ \cdots \\ >_w \\ t_{k1}\#t_{k2}\#\ldots\#t_{km_k}, \end{array} \tag{2}$$

where each $\#$ is either $=_{\mathrm{TA}}$ or $\succ$. We call a *row* in such a constraint any maximal subsequence $t_{j1}\#t_{j2}\#\ldots\#t_{jm_j}$ in which $>_w$ does not occur. So constraint (2) 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 of (2) all terms in a row have the same weight.

LEMMA 3.2 *Any refined constraint can be effectively transformed into an equivalent refined constraint that is either $\perp$, or of the form (2) and has the following property. Suppose some t_{ij} is a variable y. Then either*

1. *this variable has exactly one occurrence in rows $i,\ldots,k$, namely t_{ij} itself; or*
2. *it has two occurrences in rows $i,\ldots,k$, both in the same row: some t_{im} has the form $f(y)$ for $m < j$, and $w(f) = 0$, moreover in this case there exists at least one $\succ$ between t_{im} and t_{ij}.*

PROOF. Consider the case when t_{ij} has another occurrence.

1. *Case: this occurrence is in a term t_{iw} in the same row i.* Suppose t_{ij} occurs in t_{iw} for $w \neq j$. Consider all possible cases:

   (a) *Case: the row i contains a chain of equalities $t_{ij} =_{\text{TA}} \ldots =_{\text{TA}} t_{iw}$.* If t_{ij} is a proper subterm of t_{iw}, then the constraint has no solutions. Otherwise, t_{ij} coincides with t_{iw}, then t_{iw} can be deleted from the row, giving an equivalent constraint.

   (b) *Case: the part of row i between t_{iw} and t_{ij} contains at least one $\succ$.* Since for any solution t_{iw} and t_{ij} must have the same weight, but be distinguished by $>_{KB}$, this is possible in only one case: when $t_{iw} = f(t_{ij})$, and f has weight 0. In all other cases the constraint has no solution.

2. *Case: this occurrence is in a term t_{uv} in a different row $u > i$.* Suppose t_{ij} occurs in t_{uv} for $u > i$, then any solution would imply that the term t_{uv} has a strictly smaller weight than its subterm t_{ij}, which is impossible.

□

Note that the transformation presented in the proof of the lemma can be made in polynomial time.

We are going to prove that the constraint satisfiability problem can be reduced to the problem of satisfiability of solved form. This will be proved in Theorems 3.3 and 3.4 below. Theorem 3.3 deals with the case of the signature without unary function symbols of weight 0, in this case we can prove a stronger statement about equivalence-preserving reduction of an arbitrary constraint to a disjunction of solved forms. The presence of a unary function symbol of weight 0 introduces some complications and is treated separately in Theorem 3.4. In this case our transformation preserves a weaker form of equivalence, called the *weight equivalence* below.

THEOREM 3.3 *Let the signature Σ contain no unary function symbol of weight 0. Then any constraint C can be effectively transformed into an equivalent disjunction of solved forms.*

PROOF. We will present a required transformation. At each transformation step we assume to deal with a constraint

$$s_1 \# s_2 \# \ldots \# s_n \bigwedge y_1 = r_1 \wedge \ldots \wedge y_k = r_k \bigwedge Arith, \tag{3}$$

such that (i) $y_1 = t_1 \wedge \ldots \wedge y_k = t_k$ is in triangle form and variables y_i do not occur in $s_1, \ldots, s_n$, (ii) $s_1 \# s_2 \# \ldots \# s_n$ is refined and satisfies the conditions of Lemma 3.2.

In the beginning, using Lemma 3.2, we transform the constraint C into an equivalent constraint of the form (3), in which the triangle part is empty.

Suppose $s_1 \# s_2 \# \ldots \# s_n$ has the form

$$\begin{array}{c} t_{11} \# t_{12} \# \ldots \# t_{1m_1} \\ >_w \\ \cdots \\ >_w \\ t_{k1} \# t_{k2} \# \ldots \# t_{km_k}. \end{array} \tag{4}$$

where each $\#$ is either $=_{\text{TA}}$ or $\succ$. If all t_{ij}'s are variables and each $\#$ is $\succ$, then we are done. So to obtain a solved form, we have to show how to change all t_{ij}'s into variables. Take the topmost row i in (4) which contains at least one nonvariable. Let this row be $p_l \# p_{l-1} \# \ldots \# p_1$ (for technical reasons it is convenient for us to use indices in the decreasing order, so that p_1 is the last element of the row).

Consider two possible cases. In both cases we will turn C into an equivalent disjunction of constraints of the form (3), but each constraint in the disjunction will have less occurrences of function symbols in its refined part than C.

1. *All terms $p_1, \ldots, p_l$ in the row are nonvariables.* We will eliminate all terms from the row one by one, except for the last one. Consider the pair p_l, p_{l-1}. Since all p_i's are flat, $p_l = g(x_1, \ldots, x_u)$ and $p_{l-1} = h(y_1, \ldots, y_v)$ for some variables $x_1, \ldots, x_u, y_1, \ldots, y_v$ and $g, h \in \Sigma$. Consider two cases. In each case we give a formula F_1 equivalent to the constraint $p_l \# p_{l-1}$ and consisting only of arithmetical literals and formulas in which g, h do not occur. We will describe later what we do with this formula F_1.

   (a) *$p_l \# p_{l-1}$ is the equation $g(x_1, \ldots, x_u) = h(y_1, \ldots, y_v)$.* If $g \neq h$, then $g(x_1, \ldots, x_u) = h(y_1, \ldots, y_v) \leftrightarrow \perp$. If $g = h$ (and hence $u = v$) then $g(x_1, \ldots, x_u) = h(y_1, \ldots, y_v) \leftrightarrow x_1 = y_1 \wedge \ldots \wedge x_u = y_u$.

   (b) *$p_l \# p_{l-1}$ is $g(x_1, \ldots, x_u) \succ h(y_1, \ldots, y_v)$.* If $h \gg g$, then $g(x_1, \ldots, x_u) \succ h(y_1, \ldots, y_v)$ is equivalent to $\perp$. If $g \gg h$, then $g(x_1, \ldots, x_u) \succ h(y_1, \ldots, y_v)$ is equivalent to $|g(x_1, \ldots, x_u)| = |h(y_1, \ldots, y_v)|$. If $g = h$ (and hence $u = v$), then

$$
\begin{aligned}
& f(x_1,\dots,x_u) \succ g(y_1,\dots,y_v) \leftrightarrow \\
& \quad |g(x_1,\dots,x_u)| = |h(y_1,\dots,y_v)| \land \\
& \quad \bigvee_{i=1\dots u} (x_1 = y_1 \land \dots \land x_{i-1} = y_{i-1} \land \\
& \qquad x_i >_{KB} y_i).
\end{aligned}
$$

So in all cases we have $p_l \# p_{l-1} \equiv F_1$. It is easy to see that the constraint C is equivalent to the formula $C_1 \land F_1$, where C_1 is obtained from C by replacing the row $p_l \# p_{l-1} \# \dots \# p_1$ by $p_{l-1} \# \dots \# p_1$. We perform the same transformation on C_1, obtaining an equivalent formula $C_2 \land F_2$ where C_2 is obtained from C_1 by replacing the row $p_{l-1} \# p_{l-2} \# \dots \# p_1$ by $p_{l-2} \# \dots \# p_1$ etc. This results in a constraint C_{l-1} obtained from C by replacing $p_l \# p_{l-1} \dots \# p_1$ by p_1 and such that $C \leftrightarrow C_{l-1} \land F_1 \land \dots \land F_{l-1}$. Consider C_{l-1}, its refined part contains $s' >_w p_1 >_w s''$. Replace in C_{l-1} this formula by $s' >_w s''$, obtaining a constraint C_l and define a formula F_l to be $|s'| > |p_1| \land |p_1| > |s''|$. Evidently, C_{l-1} and $C_l \land F_l$ are equivalent.

Denote the formula $C_l \land F_1 \land \dots \land F_l$ by C' and let us summarize what we have obtained so far. First, $C \leftrightarrow C'$. Second, C_l is obtained from C by dropping the row $p_l \# p_{l-1} \# \dots \# p_1$. Third, all atoms in F_i's are either arithmetical or contain no function symbols. Therefore, the formula C' contains less function symbols than C in its refined part.

The formula C' is not a constraint of the form (3), but it can be turned into an equivalent disjunction $C'_1 \lor \dots \lor C'_w$ of such constraints as in the proof of Lemma 3.1. Then we transform each constraint C'_i as in the proof of Lemma 3.2, to guarantee that the conditions of this lemma are satisfied after this step. It is easy to see that this transformation does not increase the number of occurrences of function symbols.

2. *At least one term $p_1, \dots, p_l$ is a variable x.* Note that x has exactly one occurrence in the refined part of C. Indeed, for the row $p_l \# p_{l-1} \# \dots \# p_1$ and any row below this holds since we assume the constraint to satisfy Lemma 3.2. But all rows above this one consist of variables only, so x cannot occur in any row above. In this case we will make transformations similar to the previous case, but we will sometimes introduce new occurrences of function symbols in the current row. However, when we complete the transformation on the row, all newly introduced occurrences of function symbols will disappear from the refined part, as well as all function symbols that previously occurred in $p_l \# p_{l-1} \dots \# p_1$.

   Take any i such that p_i is a variable x. By our assumption, it has exactly one occurrence in the refined part. Consider the following valid formula G

$$\bigvee_{g \in \Sigma} \exists \bar{y} (x = g(\bar{y})). \tag{5}$$

   Since C is equivalent to $C \land G$, we can turn C into an equivalent disjunction of formulas $x = g(\bar{y}) \land C$. Replace x by $g(\bar{y})$ in the refined part of C, obtaining a constraint C', then $x = g(\bar{y})$ can be considered as belonging to the triangle part of C'. Since x had only one occurrence in C, the only difference between the refined parts of C and C' is that x is replaced by $g(\bar{y})$ in the row $p_l \# p_{l-1} \# \dots \# p_1$.

   We do this transformation for all p_i in the row that are variables. After this, the row will only contain nonvariable terms, and we do the same as in case 1.

   In this case we temporarily introduced new occurrences of function symbols in the row, but then eliminated all of them, and in addition at least one occurrence of a function symbol that was in the row initially. Thus, the number of occurrences of function symbols in the refined part decreases.

Since every transformation step decreases the number of occurrences of function symbols in the refined part, we will eventually obtain a disjunction of constraints with the refined part of the form (4), but without function symbols. Suppose we have a constraint $x_i =_{\mathrm{TA}} x_j$ in the refined part. Then we substitute x_i instead of all occurrences of x_j in the constraint and add $x_j =_{\mathrm{TA}} x_i$ in the triangle part, obtaining an equivalent constraint. Thus we can get rid of $=_{\mathrm{TA}}$ in the refined part and obtain a solved form. □

Note that the transformation presented in this theorem cannot be presented as a nondeterministic polynomial-time algorithm, since there is no obvious polynomial bound on the number of new variables introduced by the transformation. However, there is an obvious exponential bound, so the transformation can be presented as a nondeterministic exponential-time algorithm. This is also true for the algorithm presented in the next theorem.

Let now the signature Σ contain a unary function symbol f of weight 0. Let us introduce several definitions. We 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 f-terms have positive f-height, while non f-terms have f-height 0. 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.

A substitution θ is called *grounding* for a set or sequence of variables X is for every x in X the term $\theta(x)$ is ground. Two substitutions θ_1 and θ_2 are called *weight-equivalent* with respect to a set of variables X if they are grounding for X, and for every $x \in X$ the weight of $x\theta_1$ coincides with the weight of $x\theta_2$. In the proof of Theorem 3.4 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 *weight-equivalent*. By this we mean the following.

1. For every substitution θ grounding for $\bar{x}$ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta$ there exists a substitution θ' grounding for $\bar{x}, \bar{y}$ such that $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta'$, and θ' is weight-equivalent to θ.
2. For every substitution θ' such that θ' is grounding for $\bar{x}, \bar{y}$ and $\mathrm{TA}^+(\Sigma) \models A(\bar{x}, \bar{y})\theta'$ there exists a substitution θ such that $\mathrm{TA}^+(\Sigma) \models C(\bar{x})\theta$ and θ is weight-equivalent to θ'.

So if $C(\bar{x})$ has a solution, then $A(\bar{x}, \bar{y})$ has a solution of the same weight on $\bar{x}$, and vice versa.

THEOREM 3.4 *Let the signature Σ contain a unary function symbol f of weight 0. Then any constraint C can be effectively transformed into a weight-equivalent disjunction of solved forms.*

PROOF. We present a transformation very similar to that of Theorem 3.3. At each transformation step we assume to deal with a constraint (3). As in Theorem 3.3, we use Lemma 3.2 to transform the constraint C into an equivalent constraint of the form (3), in which the triangle part is empty. We will turn C into a weight-equivalent disjunction $C_1 \vee \ldots \vee C_n$ of constraints of the form (3), so that the refined part of each C_i contains less occurrences of function symbols than C. As in Theorem 3.3, at each step we take the topmost row in constraint (4) which contains at least one nonvariable. Let this row be $p_l \# p_{l-1} \# \ldots \# p_1$. We will call it the *current row*. Unlike Theorem 3.3, it is not enough for us to consider flat terms, temporarily we can change the current row so that it contains a more general kind of terms, called *almost flat*, and then eliminate all function symbols in the row. We call a term t *almost flat* if it has the form $f^n(t)$, where (i) $n \geq 0$, (ii) t is a flat term, (iii) t is not a variable, and (iv) t is not an f-term.

We recall that each term p_i in $p_l \# p_{l-1} \# \ldots \# p_1$ is flat. Our first aim is to make them into almost flat terms. So suppose that there are terms $p_1, \ldots, p_l$ in the row that are not almost flat.

By our construction, we assume that Lemma 3.2 holds for our constraint. This means that, if some p_i is a variable x, then either x has one occurrence in the refined part of C, or it has two occurrences, both in $p_l \# p_{l-1} \# \ldots \# p_1$, the first occurrence is $f(x)$ and the second is x. Indeed, for the row $p_1 \# p_2 \# \ldots \# p_l$ and any row below this holds since we assume the constraint to satisfy Lemma 3.2. But all rows above this one consist of variables only, so again by this assumption x cannot occur in any row above.

Due to the complication introduced by the presence of f, we cannot continue exactly as in Theorem 3.3. Instead, we prove an auxiliary statement about solutions of $p_l \# p_{l-1} \# \ldots \# p_1$.

(6) If $p_l \# p_{l-1} \# \ldots \# p_1$ has a solution θ, then it has a solution θ_1 such that (i) θ_1 is weight-equivalent to 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 (6) it is enough to show the following.

(7) There exists a solution θ' such that (i) θ' is weight-equivalent to θ, (ii) the f-distance between $p_k\theta'$ and $p_{k-1}\theta'$ is $d-1$, and (iii) for every $k' \neq k$ the f-distance between $p_{k'}\theta'$ and $p_{k'-1}\theta'$ coincides with the f-distance between $p_{k'}\theta$ and $p_{k'-1}\theta$.

Let us show (7), and hence (6). Since θ is a solution to the row, that the f-distance between any $p_{k'''}\theta$ and $p_k\theta$ for $k''' \geq k$ is nonnegative. Likewise, the f-distance between any $p_{k-1}\theta$ and $p_{k''}\theta$ for $k-1 \geq k''$ 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.

(8) 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 some p_i for $i \geq k$. Then the f-height of $p_i\theta$ is at least 2. Since p_i is flat, then $p_i = f(x)$ or $p_i = x$. If x also occurs in p_j for $j < k$, since we assume that the constraint satisfies Lemma 3.2, then $p_i = f(x)$ and $p_j = x$. But then the f-distance between $p_i\theta$ and $p_j\theta$ is 1, so we obtain a contradiction. So (8) is proved.

Now note the following.

(9) If a variable x occurs in $p_{k'''}$ such that $k''' \geq k$, then $x\theta$ is a f-term.

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

Define the substitution θ' as follows:

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

By (8) and (9), θ' is defined correctly. We claim that θ' satisfies (7). The properties (i)-(iii) are straightforward by our construction, it only remains to prove that θ' is a solution of the row, i.e. for every k' we have $p_{k'}\theta' \# p_{k'-1}\theta'$. Well, for $k' > k$ we have $p_{k'}\theta = f(p_{k'}\theta')$ and $p_{k'-1}\theta = f(p_{k'-1}\theta')$, and for $k' < k$ we have $p_{k'}\theta = p_{k'}\theta'$ and $p_{k'-1}\theta = p_{k'-1}\theta'$, in both cases $p_{k'}\theta' \# p_{k'-1}\theta'$ follows from $p_{k'}\theta \# p_{k'-1}\theta$. The only difficult case is $k = k'$.

So assume $k = k'$. Since the f-distance between $p_k\theta$ and $p_{k-1}\theta$ is $d > 1$, $p_k\theta \neq p_{k-1}\theta$, and hence $p_k \# p_{k-1}$ must be $p_k \succ p_{k-1}$. Since θ is a solution of $p_k \succ p_{k-1}$ and since θ' is weight-equivalent to θ, the weights of $p_k\theta'$ and $p_{k-1}\theta'$ coincide. But then $p_k\theta' \succ p_{k-1}\theta'$ follows from the fact that the f-distance between $p_k\theta'$ and $p_{k-1}\theta'$ is $d-1 \geq 1$.

Now the proof of (7), and hence of (6), is completed. In the same way as (6), we can also prove

(10) If $p_l \# p_{l-1} \# \ldots \# p_1$ has a solution θ, then it has a solution θ_2 such that (i) θ_2 is weight-equivalent to θ, (ii) for every $1 < k \leq l$ the f-distance between $p_k\theta_1$ and $p_{k-1}\theta_2$ is at most 1. (iii) the f-height of $p_1\theta_2$ is at most 1.

From this we can derive the following statement that gives an upper bound for the (minimal) f-height of solutions.

(11) If $p_l \# p_{l-1} \# \ldots \# p_1$ has a solution θ, then it has a solution θ_2 such that (i) θ_2 is weight-equivalent to θ, (ii) for all k, if p_k is a variable x, then the f-height of $x\theta'$ is at most k, and (iii) for all k, if p_k has the form $f(x)$, then the f-height of $x\theta'$ is at most $k-1$.

Now, suppose that θ is a solution of the whole constraint C (not just the current row). Then θ also solves the current row. Change θ on the variables of the current row into a substitution θ_2 satisfying (11). It is not hard to argue that θ_2 is also a solution of the whole constraint. So we obtain

(12) If the whole constraint C has a solution θ, then it has a solution θ_2 such that (i) θ_2 is weight-equivalent to θ, (ii) for all k, if p_k is a variable x, then the f-height of $x\theta'$ is at most k, and (iii) for all k, if p_k has the form $f(x)$, then the f-height of $x\theta'$ is at most $k-1$.

Using (12), we can eliminate from the current row all terms that are not almost flat in the following way. Take any term p_k in the row that is not almost flat. Then p_k is either a variable x or a term $f(x)$. Consider the formula G defined as

$$\bigvee_{g \in \Sigma - \{f\}} \bigvee_{m=0\ldots k} \exists \bar{y}(x = f^m(g(\bar{y}))). \tag{13}$$

By (12), if the constraint C has any solution, formula $C \wedge G$ has a weight-equivalent solution. Using (13), $C \wedge G$ can be turned into an equivalent disjunction of formulas $x = f^m(g(\bar{y})) \wedge C$. Replace x by $f^m(g(\bar{y}))$ in the refined part of C, obtaining a constraint C', then $x = f^m(g(\bar{y}))$ can be considered as belonging to the triangle part of C'. Since x had at most two occurrences in C, both in the current row, the only difference between the refined parts of C and C' is that x is replaced by $f^m(g(\bar{y}))$ in the row $p_l \# p_{l-1} \# \ldots \# p_1$. We do this transformation for all terms in the row that are not almost flat.

Now, the current row only contains almost flat terms. We will eliminate all terms from the row one by one, except for the last one as in Theorem 3.3, but using almost flat terms instead of flat nonvariable terms. Consider the pair p_l, p_{l-1}. Since all p_i's are almost flat, $p_l = f^n(g(x_1, \ldots, x_u))$ and $p_{l-1} = f^m(h(y_1, \ldots, y_v))$ for some variables $x_1, \ldots, x_u, y_1, \ldots, y_v$ and function symbols $g, h \in \Sigma - \{f\}$. Consider two cases. As in Theorem 3.3, we give a formula equivalent to the constraint $p_l \# p_{l-1}$ and consisting only of arithmetical literals and formulas in which f, g, h do not occur.

1. $p_l \# p_{l-1}$ *is the equation* $f^n(g(x_1, \ldots, x_u)) = f^m(h(y_1, \ldots, y_v))$. If $m \neq n$ or $g \neq h$, then $(f^n(g(x_1, \ldots, x_u)) = f^m(h(y_1, \ldots, y_v))) \leftrightarrow \perp$. If $m = n$ and $g = h$ (and hence $u = v$) then $f^n(g(x_1, \ldots, x_u)) = f^m(h(y_1, \ldots, y_v)) \leftrightarrow x_1 = y_1 \wedge \ldots \wedge x_u = y_u$.

2. $p_l \# p_{l-1}$ *is* $f^n(g(x_1, \ldots, x_u)) \succ f^m(h(y_1, \ldots, y_v))$. If $m < n$ or ($m = n$ and $h \gg g$), then $f^n(g(x_1, \ldots, x_u)) \succ f^m(h(y_1, \ldots, y_v))$ is equivalent to $\perp$. If $m > n$ or ($m = n$ and $g \gg h$), then $f^n(g(x_1, \ldots, x_u)) \succ f^m(h(y_1, \ldots, y_v))$ is equivalent to $|g(x_1, \ldots, x_u)| = |h(y_1, \ldots, y_v)|$. If $m = n$ and $g = h$ (and hence $u = v$), then

$$\begin{aligned} f^n(g(x_1, \ldots, x_u)) \succ f^m(h(y_1, \ldots, y_v)) &\leftrightarrow \\ |g(x_1, \ldots, x_u)| = |h(y_1, \ldots, y_v)| &\wedge \\ \bigvee_{i=1\ldots u} (x_1 = y_1 \wedge \ldots \wedge x_{i-1} = y_{i-1} &\wedge \\ x_i >_{KB} y_i). \end{aligned}$$

The rest of the proof is as in Theorem 3.3. $\square$

4 From solved forms to linear Diophantine Equations

Our next aim is to show how to check satisfiability of solved forms

$$x_1\#x_2\#\ldots\#x_n \bigwedge y_1 = t_1 \wedge \ldots \wedge y_k = t_k \bigwedge Arith.$$

To obtain a solution of the whole solved form, we have to estimate for any weight x the number of terms of this weight. Moreover, for each row in the solved form

$$y_1 \succ \ldots \succ y_N$$

we have to ensure that at least N different terms of the same weight as y_1 exist.

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 as an existential formula of Presburger's Arithmetic.

We say that a relation $R(\bar{x})$ on natural numbers is $\exists$-*definable*, if there exists an existential formula of Presburger's 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 function is $\exists$-definable.

Let us fix an enumeration $g_1,\ldots,g_S$ of the signature Σ. We assume that the first B symbols $g_1,\ldots,g_B$ have arity ≥ 2, and the first F symbols $g_1,\ldots,g_F$ are nonconstants. The arity of each g_i is denoted as $arity_i$. *In this section we assume B, F, and S fixed, we also assume that N is a fixed positive integer.*

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 example, if the sequence of elements of Σ is g,h,a,b, and $t = g(h(g(g(a)),h(b,b)))$, the contents of t is $(3,2,1,2)$.

LEMMA 4.1 *The following relation $exists(x,n_1,\ldots,n_S)$ is $\exists$-definable: there exists at least one ground term of Σ with 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 equations.

The first equation is

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

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:

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

□

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

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

The second lemma implies that, if there exists any term t of weight x with at least N occurrences of function symbols, including at least two different function symbols of arity 1, then there exists at least N different terms of weight x.

LEMMA 4.3 *Let $x,n_1,\ldots,n_s$ be natural numbers such that $exists(x,n_1,\ldots,n_S)$ holds, $n_1+\ldots+n_B = 0$, $n_1+\ldots+n_F \geq N$ and at least two numbers among $n_1,\ldots,n_F$ are positive. Then there exists at least N different ground terms with the contents $(n_1,\ldots,n_S)$.* □

The following lemma is obvious but helpful.

LEMMA 4.4 *Let g be a unary function symbol, $w(g) > 0$, and c be a constant. The following function $only_{g,c}(x)$ is $\exists$-definable: $only_{g,c}(x)$ is the number of terms of weight x having the form $g^m(c)$ for some $m \geq N$.*

PROOF. Evidently, we have

$$\begin{aligned}
&only_{g,c}(x) = n \leftrightarrow\\
&\quad (n = 1 \wedge \exists m(m \geq N \wedge w(g)\cdot m + w(c) = x)) \ \vee\\
&\quad (n = 0 \wedge \exists m(m < N \wedge w(g)\cdot m + w(c) > x)) \ \vee\\
&\quad (n = 0 \wedge \exists m(\ w(g)\cdot m + w(c) < x \wedge\\
&\qquad\qquad w(g)\cdot(m+1) + w(c) > x\ )\ ).
\end{aligned}$$

□

The previous three lemmas characterize the number of terms of a given weight with at least N occurrences of function symbols, the next lemma characterizes this number for terms with less than N occurrences of function symbols.

LEMMA 4.5 *Suppose Σ is a signature with no unary function symbols of weight 0. The following function*

$small\_terms(x)$ *is* $\exists$*-definable:* $small\_terms(x)$ *is the number of terms of weight* x *having less than* N *occurrences of function symbols.*

PROOF. Note that the set T of terms with less then N occurrences of function symbols is finite and can be effectively constructed. Let W be the maximal weight of terms in T and m_k is the number of terms of the weight k in this set. Then

$$\begin{array}{l} small\_terms(x) = y \leftrightarrow \\ \quad (x = 0 \wedge y = m_0) \vee (x = 1 \wedge y = m_1) \vee \ldots \\ \quad (x = W \wedge y = m_W) \ \vee x > W \wedge y = 0. \end{array}$$

□

THEOREM 4.6 *The following relation* $at\_least_N(x)$ *is* $\exists$*-definable: there exists at least* N *terms of weight* x*.*

PROOF. First, we consider the case when our signature Σ contains a unary function symbol of weight 0. In this case it is easy to see that for every weight x, if Σ contains a term of weight x, then it contains an infinite number of such terms, therefore the formula $at\_least_N(x)$ will be

$$at\_least_N(x) = \exists n_1, \ldots, n_S exists(x, n_1, \ldots, n_S).$$

So we assume that Σ contains no unary functions of weight 0. We introduce three auxiliary relations $A_1(x), A_2(x), A_3(x)$ and write an explicit definition of $at\_least_N(x)$ as follows.

$$\begin{array}{ll} A_1(x) \leftrightarrow \exists n_1, \ldots, n_S(& exists(x, n_1, \ldots, n_S) \wedge \\ & n_1 + \ldots + n_B \geq 1 \wedge \\ & n_1 + \ldots + n_F \geq N\ ); \end{array}$$

$$\begin{array}{ll} A_2(x) \leftrightarrow \exists n_1, \ldots, n_S(& exists(x, n_1, \ldots, n_S) \wedge \\ & n_1 + \ldots + n_B = 0 \wedge \\ & n_1 + \ldots + n_F \geq N \wedge \\ & \bigvee_{B<i<j\leq F} (n_i > 0 \wedge n_j > 0)\ ); \end{array}$$

$$\begin{array}{ll} A_3(x) \leftrightarrow & small\_terms(x) + \\ & \sum_{B<i\leq F<j\leq S} only_{g_i, g_j}(x) \geq N; \end{array}$$

$$at\_least_N(x) \leftrightarrow A_1(x) \vee A_2(x) \vee A_3(x).$$

Indeed, take a positive integer x and suppose $A_1(x)$ holds. Then by Lemma 4.2, we have $at\_least_N(x)$. Likewise, if $A_2(x)$ holds, then by Lemma 4.3, we have $at\_least_N(x)$ too. If neither $A_1(x)$ nor $A_2(x)$ take place, then all terms of the weight x are either those having less than N occurrences of function symbols, plus those having the form $g^m(c)$ for some g, c and $m \geq N$. By Lemmas 4.4 and 4.5, the number of such terms is

$$small\_terms(x) + \sum_{B<i\leq F<j\leq S} only_{g_i, g_j}(x),$$

so in this case $at\_least(x)$ is equivalent to $A_3(x)$. □

We think the formula $at\_least_N(x)$ can be built in polynomial time of N, but the proof would require more involved arguments.

5 Main result

We are ready to prove Theorem 2.1: the existential first-order theory of any term algebra with a Knuth-Bendix ordering is decidable.

PROOF (of Theorem 2.1). By Proposition 2.2 it is enough to prove decidability of the constraint satisfaction problem. Take a constraint. By Theorem 3.3 it can be effectively transformed into an equivalent disjunction of solved forms, so it remains to show how to check satisfiability of a solved form.

Suppose that C is any solved form. Recall that C is of the form

$$refined \wedge triangle \wedge Arith, \tag{14}$$

such that

1. *refined* has the form

$$\begin{array}{c} x_{11} \succ x_{12} \succ x_{13} \succ \ldots \succ x_{1m_1} \\ >_w \\ \ldots \\ >_w \\ x_{i1} \succ x_{i2} \succ x_{i3} \succ \ldots \succ x_{im_i} \\ >_w \\ \ldots \\ >_w \\ x_{v1} \succ x_{v2} \succ x_{v3} \succ \ldots \succ x_{vm_v} \end{array} \tag{15}$$

   such that all variables $x_{11}, \ldots, x_{vm_v}$ are pairwise different;

2. *triangle* is a triangle form

$$y_1 = t_1 \wedge \ldots \wedge y_k = t_k,$$

   and no x_{ij} occurs in any y_w;

3. *Arith* is an arithmetical constraint.

Let $(z_1, \ldots, z_n)$ be the tuple of all variables occurring in C. Denote by *refined*$'$ the following formula

$$\bigwedge_{i=1\ldots v}\ \bigwedge_{j=1\ldots m_i} (|x_{i1}| = |x_{ij}|) \quad \wedge$$
$$\bigwedge_{1\le i<j\le v} (|x_{i1}| > |x_{j1}|) \quad \wedge \qquad (16)$$
$$\bigwedge_{i=1\ldots v} at\_least_{m_i}(|x_{i1}|).$$

Likewise, denote by $triangle'$ the following arithmetical constraint

$$|y_1| = |t_1| \wedge \ldots \wedge |y_k| = |t_k|. \qquad (17)$$

Consider the formula

$$refined' \wedge triangle' \wedge Arith \wedge \bigwedge_{i=1\ldots n} at\_least_1(|z_i|). \qquad (18)$$

First, we prove

(19) Any solution to C is also a solution to (18).

Indeed, it is not hard to argue that $refined$ implies $refined'$, since $x_{i1} \succ x_{ij}$ implies $|x_{i1}| = |x_{ij}|$ and $x_{i1} >_w x_{j1}$ implies $|x_{i1}| > |x_{j1}|$ and $x_{i1} \succ x_{i2} \succ x_{i3} \succ \ldots \succ x_{im_i}$ implies $at\_least_{m_i}(|x_{i1}|)$. Likewise, $triangle$ implies $triangle'$, since $y_i = t_i$ implies $|y_i| = |t_i|$. Evidently, for any solution we have $at\_least_1(|z_i|)$.

Recall that $(z_1, \ldots, z_n)$ is the tuple of all variables of (18), then we can speak of solutions to (18) as tuples of terms $(s_1, \ldots, s_n)$. Furthermore, we can assume that (18) contains no function symbols by repeatedly replacing each $|g(t_1, \ldots, t_n)|$ by $w(g) + |t_1| + \ldots + |t_n|$.

Introduce new arithmetical variables $z'_1, \ldots, z'_n$. Denote by $Diophantine$ the existential arithmetical formula obtained by replacing each $|z_i|$ by z'_i in (18). We can speak of solutions to $Diophantine$ as tuples of natural numbers $(w_1, \ldots, w_n)$. Then (19) yields

(20) If $(h_1, \ldots, h_n)$ is a solution to C, then $(|h_1|, \ldots, |h_n|)$ is a solution to $Diophantine$.

We leave it to the reader to check that the reverse also holds:

(21) Any solution $(w_1, \ldots, w_n)$ to $Diophantine$ can be made into a solution $(h_1, \ldots, h_n)$ to C such that $|h_i| = w_i$ for all $i = 1 \ldots n$.

Let us summarize what we have obtained so far. We reduced the decidability of the existential theory of term algebras with a Knuth-Bendix ordering to the problem of solvability of constraints of the form (14). For each such a constraint C we have effectively built an existential formula of Presburger's arithmetic $Diophantine$. (19) and (21) show that C is solvable if and only if so is $Diophantine$. It remains to use the well-known result on decidability of systems of linear Diophantine equations. □

6 Related work and open problems

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

6.1 Knuth-Bendix ordering constraints and linear Diophantine equations

The Knuth-Bedix ordering was introduced in [Knuth and Bendix 1970]. Later, [Dershowitz 1982] introduced recursive path orderings (RPOs). A number of results on recursive path orderings and solving RPO 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 orderings. We used linear Diophantine equations in our decidability proofs. Let us show that the use of linear Diophantine equations is not coincidential: they are definable in the Knuth-Bendix ordering.

EXAMPLE 6.1 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 ordering $\gg$ on Σ such that $g \gg s$. Our aim is to represent linear Diophantine equations by Knuth-Bendix constraints. To this end, we will consider any ground term t as representing the natural number $|t| - 1$.

Define the formula

$$equal\_weight(x, y) \leftrightarrow g(x) >_{KB} s(y) \wedge g(y) >_{KB} s(x).$$

It is not hard to argue that, for any ground terms r, t $equal\_weight(r, t)$ holds if and only if $|r| = |t|$.

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

$$x_1 + \ldots + x_n + k = x_0, \qquad (22)$$

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

$$equal\_weight(s^{k+2}(h(y_1, h(y_2, \ldots, h(y_{n-1}, y_n)))), s^{2n}(y_0)). \qquad (23)$$

It is not hard to argue that

(24) Formula (23) holds if and only if

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

Using (24), we can transform any system $D(x_1, \ldots, x_n)$ of linear Diophantine equations of the form (22) into a constraint $C(y_1, \ldots, y_n)$ such that for every tuple of ground terms $t_1, \ldots, t_n$, $C(t_1, \ldots, t_n)$ holds if and only if so does $D(|t_1| - 1, \ldots, |t_n| - 1)$.

6.2 The case of single inequation

Comon and Treinen [1994] proved that LPO constraint solving is NP-hard already for constraints consisting of a single inequation. Let us comment on the single inequation case for the Knuth-Bendix ordering here.

The Knuth-Bendix ordering is defined in [Knuth and Bendix 1970] also for the nonground case. If $s >_{KB} t$ for nonground terms, then $s\sigma >_{KB} t\sigma$ also holds for every substitution σ. Let us show that the Knuth-Bendix ordering for nonground terms is incomplete, i.e. there exists a Knuth-Bendix ordering $>_{KB}$ and nonground terms s, t of a signature Σ such that for every substitution σ grounding for s, t we have $s\sigma >_{KB} t\sigma$, but $s \not>_{KB} t$.

EXAMPLE 6.2 We do not define the original Knuth-Bendix ordering with variables here, the exact definitions can be found in [Knuth and Bendix 1970] or [Baader and Nipkow 1998]. Consider the following formula of one variable x:

$$g(x, a, b) >_{KB} g(b, b, a). \tag{25}$$

For any choice of the weight function and ordering $\gg$, $g(x, a, b) >_{KB} g(b, b, a)$ does not hold for the original Knuth-Bendix ordering with variables. However, formula 25 is valid in any term algebra with the Knuth-Bendix ordering where $w(a) = w(b)$ and $a \gg b$.

This example shows that the (original) Knuth-Bendix ordering with variables cannot be used for solving constraints consisting of a single inequation. In contrast to [Comon and Treinen 1994] we note

THEOREM 6.3 *There exists a polynomial-time algorithm for solving Knuth-Bendix ordering constraints consisting of a single inequation.*

The proof will be appear in [Korovin and Voronkov 2000].

6.3 Other results on ordering constraints

[Martin 1987, Dick, Kalmus and Martin 1990] consider Knuth-Bendix orderings with real-valued functions and prove sufficient and necessary conditions for a system of rewrite rules to be oriented by such an ordering. They also define an algoritm for finding orderings orienting a system of rewrite rules.

Nieuwenhuis [1993] proved NP-completeness of LPO constraint solving, Narendran, Rusinowitch and Verma [1998] proved NP-completness of RPO constraint solving. Recently, Nieuwenhuis and Rivero [1999] proposed a new effcient method for solving RPO constraints. NP-completeness of satisfiability of LPO constraints consisting of a single inequation was proved by Comon and Treinen [1994].

[Lepper 2000] studies derivation length and order types of Knuth-Bendix orderings, both for integer-valued and real-valued weight functions.

6.4 First-order theory term algebras with binary relations

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], 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 existentional theory [Venkataraman 1987]. Term algebras with lexicographic path orderings have an undecidable first-order theory [Comon and Treinen 1997].

6.5 Future work

Let us point out some directions for further work.

1. It is interesting whether there exists a nondeterministic polynomial-time algorithm for the constraint satisfiability problem.

2. It is also interesting whether the full first-order theory of term algebras with Knuth-Bendix orderings is decidable.

References

BAADER F. AND NIPKOW T. [1998], *Term Rewriting and and All That*, Cambridge University press, Cambridge.

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.

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* 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.

DERSHOWITZ N. [1982], ‘Orderings for term rewriting systems’, *Theoretical Computer Science* **17**, 279–301.

DICK J., KALMUS J. AND MARTIN U. [1990], ‘Automating the Knuth-Bendix ordering’, *Acta Informatica* **28**(2), 95–119.

HODGES W. [1993], *Model theory*, Cambridge University Press.

KIRCHNER H. [1995], On the use of constraints in automated deduction, *in* A. Podelski, ed., ‘Constraint Programming: Basics and Tools’, Vol. 910 of *Lecture Notes in Computer Science*, Springer Verlag, pp. 128–146.

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], On the use of the knuth-bendix orderings for orienting systems of rewrite rules, Preprint, Department of Computer Science, University of Manchester. To appear.

KUNEN K. [1987], ‘Negation in logic programming’, *Journal of Logic Programming* **4**, 289–308.

LEPPER I. [2000], ‘Derivations lengths and order types of Knuth-Bendix orders’, *Theoretical Computer Science* . Submitted.

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.

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.

NARENDRAN P., RUSINOWITCH M. AND VERMA R. [1998], RPO constraint solving is in NP, *in* G. Gottlob, E. Grandjean and K. Seyr, eds, ‘Computer Science Logic, 12th International Workshop, CSL’98’, 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. [1999], Rewrite-based deduction and symbolic constraints, *in* H. Ganzinger, ed., ‘Automated Deduction—CADE-16. 16th International Conference on Automated Deduction’, Lecture Notes in Artificial Intelligence, Trento, Italy, pp. 302–313.

NIEUWENHUIS R. AND RIVERO J. [1999], Solved forms for path ordering constraints, *in* ‘In Proc. 10th International Conference on Rewriting Techniques and Applications (RTA)’, Vol. 1631 of *Lecture Notes in Computer Science*, Trento, Italy, pp. 1–15.

RYAZANOV A. AND VORONKOV A. [1999], Vampire, *in* H. Ganzinger, ed., ‘Automated Deduction—CADE-16. 16th International Conference on Automated Deduction’, Lecture Notes in Artificial Intelligence, Trento, Italy, pp. 292–296.

SUTCLIFFE G. [2000], ‘The CADE-16 ATP system competition’, *Journal of Automated Reasoning* . to appear.

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.

WEIDENBACH C. [1999], Combining superposition, sorts and splitting, *in* A. Robinson and A. Voronkov, eds, ‘Handbook of Automated Reasoning’, Elsevier Science and MIT Press. To appear.