The decidability of the first-order theory of the Knuth-Bendix order in the case of unary signatures

Konstantin Korovin and Andrei Voronkov

University of Manchester
{korovin,voronkov}@cs.man.ac.uk

Abstract. We show that the first-order theory of any Knuth-Bendix order in the case of the signatures consisting of unary function symbols and constants is decidable. Our decision procedure uses interpretation of unary terms as trees and uses decidability of the weak monadic second-order theory of binary trees. One area of applications of our result is automated deduction, since using the first-order theory of the Knuth-Bendix orders we can decide an important class of ordering constraints.

1 Introduction

Introduction of ordering constraints has been one of the main breakthroughs in the saturation based theorem proving. Using solvability of ordering constraints we can dramatically reduce the number of redundant inferences in a resolution-based prover. As a consequence, the problem of solving ordering constraints for the known simplification orders is one of the important problems in the area. A simplification order is a total monotonic order on ground terms. Given such an order we can consider ordering constraints which are quantifier-free formulas in the language of the term algebra with equality and the order. Two kinds of orders are mainly used in automated deduction: the Knuth-Bendix orders [9] and various versions of the recursive path orders [5]. Because of its importance, the decision problem for ordering constraints has been well-studied. For the recursive path orders decidability and complexity issues were considered in [8, 2, 16, 17, 15, 14]. For the Knuth-Bendix orders we have the following results: the decidability of constraints [10], a nondeterministic polynomial-time algorithm for constraint solving [11], a polynomial-time algorithm for solving constraints consisting of a single inequality [12].

In resolution-based theorem proving there are important simplifications which allow us to remove clauses form the search space (for example subsumption). It turns out that in order to express applicability conditions for these simplifications, we need to consider constraints which involve first-order quantifiers. Unfortunately the first-order theory of the recursive path orders is undecidable [20, 4]. Only recently the decidability of the first-order theory of recursive path orders in the case of unary signatures has been proven [14]. A signature is called unary if it consists of unary function symbols and constants.

In this paper we prove the decidability of the first-order theory of the Knuth-Bendix orders in the case of unary signatures. Our decision procedure uses interpretation of unary terms as trees and uses decidability of the weak monadic second-order theory of binary trees.

2 Preliminaries

A *signature* is a finite set of function symbols with associated arities. *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 $g^{\mathrm{TA}(\Sigma)}(t_1, \ldots, t_n) = g(t_1, \ldots, t_n)$. For details see e.g. [7] or [13]. It is easy to see that in term algebras any ground term is interpreted by itself. The Knuth-Bendix order is a family of orders parametrized by two parameters: a weight function and a precedence relation.

DEFINITION 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 exist at most one unary function symbol $f \in \Sigma$ such that $w(f) = 0$. Given a weight function w, we call $w(g)$ the *weight* of g. The *weight* of any ground term t, denoted $|t|$, is defined as follows: for every constant c we have $|c| = w(c)$ and for every function symbol g of a positive arity we have $|g(t_1, \ldots, t_n)| = w(g) + |t_1| + \ldots + |t_n|$.

These conditions on the weight function ensure that the Knuth-Bendix order is a simplification order total on ground terms (see, e.g., [1]). In this paper, f *will always denote a unary function symbol of weight* 0.

DEFINITION 2 (precedence relation) 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, whenever f is a unary function symbol f of weight zero, f is the greatest element w.r.t. $\gg$.

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

DEFINITION 3 The *Knuth-Bendix order* on $\mathrm{TA}(\Sigma)$ is the binary relation $\succ$ defined as follows. For any ground terms $t = g(t_1, \ldots, t_n)$ and $s = h(s_1, \ldots, s_k)$ we have $t \succ s$ 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 $t_1 = s_1, \ldots, t_{i-1} = s_{i-1}$ and $t_i \succ s_i$.

Note that the Knuth-Bendix order is a total monotonic well-founded order, see, e.g., [1]. Let $\mathrm{TA}_{\succ}(\Sigma)$ denote the structure of the term algebra over Σ with the Knuth-Bendix order $\succ$.

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

3 Interpretations

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

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

Let us introduce a simple notion of interpretation which we will use later to show the decidability of the first-order theory of the unary Knuth-Bendix orders. For a more general theory of interpretations see, e.g., [7, 6, 18]. In the sequel we will use lower-case letters $x, y, z, \ldots$ to denote first-order variables and upper-case letters $X, Y, Z, \ldots$ to denote second-order variables.

Definition 4 Let A be a structure in a first-order language L_A and B be a structure in a weak monadic second-order language L_B. We say that the structure A is interpretable in the structure B if there exist a positive integer m and the folling formulas:

1. $\phi_{domain}(\bar{X})$, where $\bar{X}$ is a tuple of second-order variables of the length m such that the set $A' = \{\bar{S} \mid B \models \phi_{domain}(\bar{S})\}$ is non-empty;
2. $\phi_g(\bar{X}_1, \ldots, \bar{X}_n, \bar{Y})$ for each function symbol g in the language L_A, where the arity of g is n and $\bar{X}_1, \ldots, \bar{X}_n, \bar{Y}$ are tuples of second-order variables of the length m, and this formula defines a function, denoted by g', on A', i.e., we have
$$g'(\bar{S}_1, \ldots, \bar{S}_n) = \bar{T} \Leftrightarrow B \models \phi_g(\bar{S}_1, \ldots, \bar{S}_n, \bar{T});$$
3. $\phi_P(\bar{X}_1, \ldots, \bar{X}_n)$ for each predicate symbol P in L_A, where the arity of P is n and $\bar{X}_1, \ldots, \bar{X}_n$ are tuples of second-order variables of the length m, and this formula defines a predicate on A', denoted by P', i.e., we have
$$P'(\bar{S}_1, \ldots, \bar{S}_n) \Leftrightarrow B \models \phi_P(\bar{S}_1, \ldots, \bar{S}_n);$$

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

We will use the following fundamental property of interpretability.

Proposition 1. *If a structure A is interpretable in the structure B and the theory of B (in the language L_B) is decidable, then the theory of A (in the language L_A) is also decidable.*

The proof can be found, e.g. in [7, 6, 18].

4 Interpretation of the Knuth-Bendix order in WS2S

We will use interpretations to show the decidability of the first-order theory of the unary Knuth-Bendix orders. We show how to interpret Knuth-Bendix orders in the structure of two successors in the weak monadic language. Then, using the result [19] on the decidability of the weak monadic theory of two successors, we conclude that the first-order theory of the unary Knuth-Bendix orders is decidable.

Let us briefly recall the definition of the structure of two successors (see, e.g., [3] for details). The domain consists of finite binary strings including the empty string λ. There are two functions $0(x)$ and $1(x)$ which add 0 and 1 respectively to the end of the string. For example $0(101) = 1010$. Instead of $0(t)$ and $1(t)$ we will write, respectively, $t \cdot 0$ and $t \cdot 1$. The atomic formulas are equalities $t = s$ between first-order terms, and $t \in X$ where t is a first-order term. Formulas are built from atomic formulas using logical connectives $\wedge, \vee, \neg$, the first-order quantifiers $\exists x, \forall x$ and second-order quantifiers over finite sets $\exists X, \forall X$. We will use the following standard shorthands: $\exists x \in X \phi(x, X)$ for $\exists x(x \in X \wedge \phi(x, X))$ and $\forall x \in X \phi(x, X)$ for $\forall x(x \in X \supset \phi(x, X))$. Binary strings can be seen as positions in binary trees, and in the sequel we sometimes will use the word *position* instead of string.

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

Emptiness:

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

Intersection:

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

Union:

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

Partition:

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

PrefixClosed:

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

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

Prefix order $\sqsubseteq$:

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

Likewise, we introduce

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

Lexicographic order $\leq_{lex}$:

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

Likewise, we introduce

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

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

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

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

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

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

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

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

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

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

Coding of terms.

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

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

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

Before giving a formal definition of the representation of terms as labelled trees, let us consider trees as sets of binary strings. Any binary tree without labels can be defined as a set of binary strings, namely the positions of the nodes in the tree. For example, the tree of Figure 1 contains the binary strings λ labelled with c, strings 1 and 11 labelled as g, string 111 labelled by h, and strings 110, 1100, and 1110 labelled by f.

Formally, for each term t we define a labelled binary tree Tree_t and two positions Right_t and Top_t in this tree. The definition is by induction on t.

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

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

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

Now we will show how to represent labelled trees by $s+1$-tuples. Let T be a labelled tree whose set of positions is X. Then we represent T as the tuple

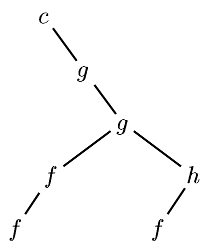

Fig. 1. The labelled tree representation of $fhffgc$, $w(f) = 0$, $w(g) = 2$, $w(h) = w(c) = 1$

$\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$, where each set X_{g_i} is the set of positions labelled by g_i and X is the set of all positions in this tree. If a term t is represented by a labelled tree T, and T is represented by a tuple $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$, we will also say that the tuple $\langle X, X_{g_1}, \ldots, X_{g_s} \rangle$ represents the term t.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

1-set:

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

n-interval:

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

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

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

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

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

Term:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Knuth-Bendix order:

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

LEMMA 5 *The formulas* $\mathit{Term}(\bar{X}), \bar{X} \succ \bar{Y}$ *and* $\phi_g(\bar{X}, \bar{Y})$ *for* $g \in \Sigma$, *define an interpretation of the term algebra with the Knuth-Bendix order in the structure of two successors.*

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

Using the decidability of the weak monadic second-order theory of two successors, this lemma and Proposition 1 we obtain the main result of this paper.

THEOREM 6 *The first-order theory of any Knuth-Bendix order in the case of the unary signatures is decidable.*

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

5 Open problems

Let us mention some open problems. The first problem is to investigate the complexity of first-order theories of Knuth-Bendix orders in the case of unary signatures. Our algorithm uses decidability of the weak monadic second-order theory of two successors, which is known to be reasonably efficient in practice, but the complexity of this theory is non-elementary.

Another open problem is the decidability of the first-order theory of Knuth-Bendix orders in the case of arbitrary signatures.

6 Acknowledgment

The authors are grateful to the anonymous referees for helpful comments.

References

1. F. Baader and T. Nipkow. *Term Rewriting and and All That.* Cambridge University press, Cambridge, 1998.
2. H. Comon. Solving symbolic ordering constraints. *International Journal of Foundations of Computer Science*, 1(4):387–411, 1990.
3. H. Comon, M. Dauchet, R. Gilleron, F. Jacquemard, D. Lugiez, S. Tison, and M. Tommasi. Tree automata techniques and applications. Available on: http://www.grappa.univ-lille3.fr/tata, 1997.
4. H. Comon and R. Treinen. The first-order theory of lexicographic path orderings is undecidable. *Theoretical Computer Science*, 176(1-2):67–87, 1997.
5. N. Dershowitz. Orderings for term rewriting systems. *Theoretical Computer Science*, 17:279–301, 1982.
6. Yuri Ershov. *Problems of decidability and constructive models.* Nauka, 1980. (in Russian).
7. W. Hodges. *Model theory.* Cambridge University Press, 1993.
8. J.-P. Jouannaud and M. Okada. Satisfiability of systems of ordinal notations with the subterm property is decidable. In *Automata, Languages and Programming, 18th International Colloquium (ICALP)*, volume 510 of *Lecture Notes in Computer Science*, pages 455–468, 1991.
9. D. Knuth and P. Bendix. Simple word problems in universal algebras. In J. Leech, editor, *Computational Problems in Abstract Algebra*, pages 263–297. Pergamon Press, Oxford, 1970.
10. K. Korovin and A. Voronkov. A decision procedure for the existential theory of term algebras with the Knuth-Bendix ordering. In *Proc. 15th Annual IEEE Symp. on Logic in Computer Science*, pages 291–302, Santa Barbara, California, June 2000.
11. K. Korovin and A. Voronkov. Knuth-bendix ordering constraint solving is NP-complete. In F. Orejas, P.G. Spirakis, and J. van Leeuwen, editors, *Automata, Languages and Programming, 28th International Colloquium, ICALP 2001*, volume 2076 of *Lecture Notes in Computer Science*, pages 979–992. Springer Verlag, 2001.
12. K. Korovin and A. Voronkov. Verifying orientability of rewrite rules using the Knuth-Bendix order. In A. Middeldorp, editor, *Rewriting Techniques and Applications, 12th International Conference, RTA 2001*, volume 2051 of *Lecture Notes in Computer Science*, pages 137–153. Springer Verlag, 2001.

13. M. Maher. Complete axiomatizations of the algebras of finite, rational and infi nite trees. In *Proc. IEEE Conference on Logic in Computer Science (LICS)*, pages 348–357, 1988.
14. Narendran and Rusinowitch. The theory of total unary rpo is decidable. In *First International Conference on Computational Logic*, volume 1861 of *Lecture Notes in Computer Science*, pages 660–672, 2000.
15. Narendran, Rusinowitch, and Verma. RPO constraint solving is in NP. In *CSL: 12th Workshop on Computer Science Logic*, volume 1584, pages 385–398. LNCS, Springer-Verlag, 1998.
16. R. Nieuwenhuis. Simple LPO constraint solving methods. *Information Processing Letters*, 47:65–69, 1993.
17. R. Nieuwenhuis and J.M. Rivero. Solved forms for path ordering constraints. In *In Proc. 10th International Conference on Rewriting Techniques and Applications (RTA)*, volume 1631 of *Lecture Notes in Computer Science*, pages 1–15, Trento, Italy, 1999.
18. Michael O. Rabin. Decidable theories. In J. Barwise, editor, *Handbook of Mathematical Logic*, volume 90 of *Studies in Logic and the Foundations of Mathematics*, chapter C.3, pages 595–629. North-Holland, Amsterdam, 1977.
19. James W. Thatcher and Jesse B. Wright. Generalized finite automata theory with an application to a decision problem of second-order logic. *Mathematical Systems Theory*, 2(1):57–81, 1968.
20. Ralf Treinen. A new method for undecidability proofs of first order theories. In *Foundations of Software Technology and Theoretical Computer Science,(FSTTCS), 10th conference*, volume 472 of *Lecture Notes in Computer Science*, pages 48–62, 1990. (journal version [21]).
21. Ralf Treinen. A new method for undecidablity proofs of first order theories. *Journal of Symbolic Computations*, 14(5):437–458, 1992.