Inclusion Between the Frontier Language of a Non-Deterministic Recursive Program Scheme and the Dyck Language is Undecidable

Naoki Kobayashi

Department of Computer Science, The University of Tokyo, Japan

Abstract

In 1970's, Nivat studied recursive program schemes (a.k.a order-1 higher-order recursion schemes in modern terminology), first-order tree grammars for generating possibly infinite trees. We consider the inclusion problem between the frontier language of a non-deterministic recursive program scheme (equivalently, an order-2 word language or indexed language) and the Dyck language, and prove that it is undecidable by a reduction from the undecidability of Hilbert's 10th problem. Essentially the same result has recently been proved by Uezato and Minamide, but our proof is arguably more direct, demonstrating the expressive power of higher-order grammars.

Keywords: recursive program schemes, higher-order languages, Dyck language, Diophantine equations

1. Introduction

In 1970's, Nivat et al. [1, 2, 3, 4] studied *recursive program schemes*, first-order tree grammars that generate possibly infinite trees. They were later extended to higher-order ones (called higher-order recursion schemes or HORS in short, in modern terminology) [5, 6], and associated model checking problems (of checking whether the trees generated by HORS satisfy a given property) have been studied recently [7, 8, 9]. The studies on HORS model checking further laid a foundation for automated verification of higher-order programs [10, 11, 9, 12].

From a language-theoretic point of view (cf. the automata-theoretic approach to model checking [13, 14]), the HORS model checking problem may

Email address: `koba@is.s.u-tokyo.ac.jp` (Naoki Kobayashi) be viewed as the inclusion problem (or the membership problem, if HORS is deterministic) between higher-order languages and (ω-)regular word/tree languages. Since finite state model checking problems may be viewed as the inclusion problem between ω-regular languages, HORS model checking may be considered an extension of finite state model checking, where the lefthand side of the language inclusion has been extended from regular (i.e. order-0) to higher-order languages. In contrast, extending the righthand side has been considered difficult. In fact, even the inclusion between regular and context-free languages is known to be undecidable [15]. A few positive results exist, however: the inclusion between context-free and superdeterministic languages is decidable [16]. Furthermore, as a special case, the inclusion between context-free languages and the Dyck language can be decided in polynomial time [17, 18]. A natural question is, therefore, whether these positive results can be extended to the case where the lefthand side is a higher-order language.

In the present article, we give a negative answer to the question above. More precisely, the inclusion problem: $L \overset{?}{\subseteq} \mathbf{D}$ is undecidable, where L is an order-2 word language (equivalently, the frontier language of a non-deterministic recursive program scheme) [6] and $\mathbf{D}$ is the Dyck language. The proof is based on a reduction from the undecidability of Hilbert's 10th problem (i.e., unsolvability of Diophantine equations) [19]. Incidentally, the Dyck language was also the subject of studies of Nivat [20]. Since the class of order-2 word languages coincides with that of indexed languages [21], our result is essentially the same as that of Uezato and Minamide [22] on the undecidability of the inclusion between indexed languages and the Dyck language. Our proof is, however, arguably more direct.[1]

The rest of this article is structured as follows. Section 2 reviews basic definitions. Sections 3 and 4 prove the main result stated above. Section 5 discusses related work and Section 6 concludes the article.

2. Preliminaries

We recall below the definition of (unsafe) non-deterministic higher-order recursion schemes [7, 8] as generators of word/tree languages, and the relationship with relevant notions, such as recursive program schemes studied

[1]Actually, our result was announced earlier than theirs [22] through the unpublished manuscript titled "Balancedness of the Words Generated by a Recursion Scheme is Undecidable", cited by [9]. The present paper is a revised version of the manuscript.

by Nivat et al. [1, 2, 3, 4] and high-level languages studied by Damm [6].

We write $\mathit{dom}(f)$ for the domain of a map f. For two maps f and g such that $\mathit{dom}(f) \cap \mathit{dom}(g) = \emptyset$, we write $f \cup g$ for the map h such that $\mathit{dom}(h) = \mathit{dom}(f) \cup \mathit{dom}(g)$, $h(x) = f(x)$ for each $x \in \mathit{dom}(f)$, and $h(x) = g(x)$ for each $x \in \mathit{dom}(g)$. We write $\{x_1 \mapsto c_1, \dots, x_k \mapsto c_k\}$ for the map f such that $\mathit{dom}(f) = \{x_1, \dots, x_k\}$ and $f(x_i) = c_i$ for each $i \in \{1, \dots, k\}$.

Let $\mathcal{A}$ be a *ranked alphabet*, i.e., a map from a finite set of symbols to the set $\mathbf{Nat}$ of natural numbers. For a symbol $a \in \mathit{dom}(\mathcal{A})$, $\mathcal{A}(a)$ is called the *arity* of a. The set of $\mathcal{A}$-labeled trees, denoted by $\mathbf{Tree}_{\mathcal{A}}$, is inductively defined by the rule: $a\, T_1 \cdots T_k \in \mathbf{Tree}_{\mathcal{A}}$ if $\mathcal{A}(a) = k$ and $T_1, \dots, T_k \in \mathbf{Tree}_{\mathcal{A}}$ (note that k may be 0; thus, $a \in \mathbf{Tree}_{\mathcal{A}}$ if $\mathcal{A}(a) = 0$).

The set of *types* is given by:

$$\kappa ::= \mathsf{o} \mid \kappa_1 \to \kappa_2.$$

Intuitively, the type o describes trees (or words, represented as unary trees), and the type $\kappa_1 \to \kappa_2$ describes functions that take an element of type κ_1 and return an element of type κ_2. The *order* of a type κ, written $\mathit{order}(\kappa)$ is defined by:

$$\mathit{order}(\mathsf{o}) = 0 \qquad \mathit{order}(\kappa_1 \to \kappa_2) = \max(\mathit{order}(\kappa_1) + 1, \kappa_2).$$

As usual, we assume that $\to$ is right-associative, so that $\kappa_1 \to \kappa_2 \to \kappa_3$ means $\kappa_1 \to (\kappa_2 \to \kappa_3)$. We often write $\kappa^n \to \kappa'$ for $\underbrace{\kappa \to \cdots \kappa}_{n} \to \kappa'$. A *type environment* is a map from a finite set of symbols to types. For a ranked alphabet $\mathcal{A}$, we write $\mathcal{K}_{\mathcal{A}}$ for the type environment $\{a \mapsto \mathsf{o}^{\mathcal{A}(a)} \to \mathsf{o} \mid a \in \mathit{dom}(\mathcal{A})\}$. The set of *applicative terms of type κ under a type environment $\mathcal{K}$*, written $\mathbf{ATerms}_{\mathcal{K},\kappa}$, is defined inductively by: (i) $x \in \mathbf{ATerms}_{\mathcal{K},\kappa}$ if $\mathcal{K}(x) = \kappa$, and (ii) $t_1 t_2 \in \mathbf{ATerms}_{\mathcal{K},\kappa}$ if $t_1 \in \mathbf{ATerms}_{\mathcal{K},\kappa' \to \kappa}$ and $t_2 \in \mathbf{ATerms}_{\mathcal{K},\kappa'}$. When $t \in \mathbf{ATerms}_{\mathcal{K} \cup \{x_1 \mapsto \kappa_1, \dots, x_k \mapsto \kappa_k\},\kappa}$ and $t_i \in \mathbf{ATerms}_{\mathcal{K},\kappa_i}$ for each $i \in \{1, \dots, k\}$, we write $[t_1/x_1, \dots, t_k/x_k]t \in \mathbf{ATerms}_{\mathcal{K},\kappa}$ for the applicative term obtained from t by simultaneously replacing each occurrence of x_i with t_i. When $t_1 \in \mathbf{ATerms}_{\mathcal{K},\kappa \to \kappa}$ and $t_2 \in \mathbf{ATerms}_{\mathcal{K},\kappa}$, we often write $t_1^{\ell} t_2$ for the term $\underbrace{t_1(\cdots(t_1}_{\ell}\, t_2)\cdots)$. We assume that there exists a countably infinite set $\mathbf{V}$ of variables.

Definition 1. A *(non-deterministic) higher-order recursion scheme (HORS, for short)* $\mathcal{G}$ is a quadruple $(\mathcal{A}, \mathcal{N}, \mathcal{R}, S)$, where

- $\mathcal{A}$ is a ranked alphabet. We call an element of $dom(\mathcal{A})$ a *terminal* symbol.

- $\mathcal{N}$ is a map from a finite set of symbols called *non-terminals* to types. We use metavariables $F, G, \ldots$ for non-terminals.

- $\mathcal{R}$ is a set of rewrite rules of the form $F\, x_1 \, \cdots \, x_k \to t$, where $x_1, \ldots, x_k \in \mathbf{V}$, $\mathcal{N}(F) = \kappa_1 \to \cdots \to \kappa_k \to \mathtt{o}$, and $t \in \mathbf{ATerms}_{\mathcal{K}_\mathcal{A} \cup \mathcal{N} \cup \{x_1 \mapsto \kappa_1, \ldots, x_k \mapsto \kappa_k\}, \mathtt{o}}$ for some $\kappa_1, \ldots, \kappa_k$.

- S, called the *start symbol*, is a non-terminal such that $\mathcal{N}(S) = \mathtt{o}$.

The *order* of a HORS is the largest order of the types of non-terminals.

A rewriting relation $t \longrightarrow_\mathcal{G} t'$, where $t, t' \in \mathbf{ATerms}_{\mathcal{K}_\mathcal{A} \cup \mathcal{N}, \mathtt{o}}$, is defined by: (i) $F\, t_1 \, \cdots \, t_k \longrightarrow_\mathcal{G} [t_1/x_1, \ldots, t_k/x_k]t$ if $F\, x_1 \, \cdots \, x_k \to t \in \mathcal{R}$; and (ii) $a\, t_1 \, \cdots \, t_i \, \cdots \, t_k \longrightarrow_\mathcal{G} a\, t_1 \, \cdots \, t'_i \, \cdots \, t_k$ if $t_i \longrightarrow_\mathcal{G} t'_i$.

The *tree language* of $\mathcal{G}$, written by $\mathcal{TL}(\mathcal{G})$, is defined by:

$$\mathcal{TL}(\mathcal{G}) = \{T \in \mathbf{Tree}_\mathcal{A} \mid S \longrightarrow^*_\mathcal{G} T\}.$$

Given a HORS $\mathcal{G}$, we sometimes refer to the four components of $\mathcal{G}$ by $\mathcal{A}_\mathcal{G}$, $\mathcal{N}_\mathcal{G}$, $\mathcal{R}_\mathcal{G}$, and $S_\mathcal{G}$. An order-1 HORS is just a non-deterministic recursive program scheme studied by Arnold and Nivat [3] (though they considered non-deterministic recursive program schemes also as generators of *infinite* trees; see Section 5).

In the present article, we are only interested in HORS's as generators of word languages. There are two ways to define the word language generated by a HORS [6]. One is via the so called *front* operation [6]. For a tree T, we define $\mathtt{front}(T) \in \{a \mid \mathcal{A}(a) = 0\}^*$ inductively by: (i) $\mathtt{front}(a) = a$ if $\mathcal{A}(a) = 0$, and (ii) $\mathtt{front}(a\, T_1 \, \cdots \, T_k) = \mathtt{front}(T_1) \cdots \mathtt{front}(T_k)$. Then, the *frontier language* of $\mathcal{G}$, written $\mathcal{FL}(\mathcal{G})$, is defined by:

$$\mathcal{FL}(\mathcal{G}) = \{\mathtt{front}(T) \mid T \in \mathcal{TL}(\mathcal{G})\}.$$

The other way is to consider paths. When the arity of each terminal symbol of a HORS $\mathcal{G}$ is at most 1, for each (unary) tree $T \in \mathbf{Tree}_{\mathcal{A}_\mathcal{G}}$, we define $\mathtt{path}(T)$ inductively by: (i) $\mathtt{path}(a) = \epsilon$ if $\mathcal{A}(a) = 0$ and (ii) $\mathtt{path}(a\, T) = a\, \mathtt{path}(T)$ if $\mathcal{A}(a) = 1$. The *word (or path) language* generated by $\mathcal{G}$ is defined by:

$$\mathcal{L}(\mathcal{G}) = \{\mathtt{path}(T) \mid T \in \mathcal{TL}(\mathcal{G})\}.$$

Henceforth, we call the word language generated by an order-k HORS *an order-k word language.*

Example 1. Consider the order-1 HORS $\mathcal{G}_1 = (\{\mathtt{br} \mapsto 2, \mathtt{a} \mapsto 0, \mathtt{b} \mapsto 0, \mathtt{c} \mapsto 0\}, \mathcal{N}, \mathcal{R}, S)$ where

$$\begin{aligned}
&\mathcal{N} = \{S \mapsto \mathtt{o}, F : \mathtt{o} \to \mathtt{o} \to \mathtt{o} \to \mathtt{o}\}\\
&\mathcal{R} = \{S \to F\,\mathtt{a}\,\mathtt{b}\,\mathtt{c},\\
&\qquad F\,x\,y\,z \to \mathtt{br}\,x\,(\mathtt{br}\,y\,z),\\
&\qquad F\,x\,y\,z \to F\,(\mathtt{br}\,\mathtt{a}\,x)\,(\mathtt{br}\,\mathtt{b}\,y)\,(\mathtt{br}\,\mathtt{c}\,z)\}.
\end{aligned}$$

We can rewrite S as follows.

$$\begin{aligned}
&S \to F\,\mathtt{a}\,\mathtt{b}\,\mathtt{c} \to^* F\,((\mathtt{br}\,\mathtt{a})^{m-1}\mathtt{a})\,((\mathtt{br}\,\mathtt{b})^{m-1}\mathtt{b})\,((\mathtt{br}\,\mathtt{c})^{m-1}\mathtt{c})\\
&\to \mathtt{br}\,((\mathtt{br}\,\mathtt{a})^{m-1}\mathtt{a})\,(\mathtt{br}\,((\mathtt{br}\,\mathtt{b})^{m-1}\mathtt{b})\,((\mathtt{br}\,\mathtt{c})^{m-1}\mathtt{c})).
\end{aligned}$$

Thus, $\mathcal{FL}(\mathcal{G}_1) = \{\mathtt{a}^m\mathtt{b}^m\mathtt{c}^m \mid m \geq 1\}$. □

Example 2. Consider the order-2 HORS $\mathcal{G}_2 = (\{\mathtt{a} \mapsto 1, \mathtt{b} \mapsto 1, \mathtt{c} \mapsto 1, \mathtt{e} \mapsto 0\}, \mathcal{N}, \mathcal{R}, S)$ where

$$\begin{aligned}
&\mathcal{N} = \{S \mapsto \mathtt{o}, F : (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to \mathtt{o},\\
&\qquad C : (\mathtt{o} \to \mathtt{o}) \to (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \to \mathtt{o}\}\\
&\mathcal{R} = \{S \to F\,\mathtt{a}\,\mathtt{b}\,\mathtt{c},\\
&\qquad F\,x\,y\,z \to x(y(z\,\mathtt{e})),\\
&\qquad F\,x\,y\,z \to F\,(C\,\mathtt{a}\,x)\,(C\,\mathtt{b}\,y)\,(C\,\mathtt{c}\,z),\\
&\qquad C\,x\,y\,w \to x(y\,w)\}.
\end{aligned}$$

We can rewrite S as follows.

$$\begin{aligned}
&S \to F\,\mathtt{a}\,\mathtt{b}\,\mathtt{c} \to^* F\,((C\,\mathtt{a})^{m-1}\mathtt{a})\,((C\,\mathtt{b})^{m-1}\mathtt{b})\,((C\,\mathtt{c})^{m-1}\mathtt{c})\\
&\to ((C\,\mathtt{a})^{m-1}\mathtt{a})(((C\,\mathtt{b})^{m-1}\mathtt{b})\,(((C\,\mathtt{c})^{m-1}\mathtt{c})\mathtt{e}))\\
&\to^* \mathtt{a}^m(\mathtt{b}^m(\mathtt{c}^m\,\mathtt{e})).
\end{aligned}$$

Thus, $\mathcal{L}(\mathcal{G}_2) = \{\mathtt{a}^m\mathtt{b}^m\mathtt{c}^m \mid m \geq 1\}$. □

We summarize below known relationships between order-k word languages and other notions of word languages.

- The class of order-k word languages includes that of Damm's level-k OI languages [6]. In fact, the definition is almost identical except the subtle condition on "safety" (we do not discuss the safety condition here; see, e.g., [23]). The two classes coincide [24] up to order-2, but it is open whether the inclusion is strict for order-3 or higher.

- The classes of order-0, order-1, and order-2 word languages respectively coincide with those of regular, context-free, and indexed languages [6, 25].

- For $k \geq 1$, the class of order-k word languages that do not contain the empty word ϵ coincides with the class of frontier languages of order-$(k-1)$ HORS's [6, 26].[2] In particular, the class of order-2 word languages that do not contain ϵ coincides with the class of frontier languages of non-deterministic recursive program schemes. Furthermore, given an order-k HORS $\mathcal{G}$ with a ranked alphabet consisting of symbols of arity at most 1, an order-$(k-1)$ HORS $\mathcal{G}'$ such that $\mathcal{FL}(\mathcal{G}') = \mathcal{L}(\mathcal{G}) \setminus \{\epsilon\}$ can be effectively constructed, and vice versa [26].

We consider the inclusion problem between an order-k word language and the Dyck language $\mathbf{D}$, i.e., the set of well-bracketed words. In this article, we write $\mathtt{a}$ and $\mathtt{b}$ for left and right brackets.

Definition 2 (Dyck language). *We write $\#_{\mathtt{a}}(w)$ ($\#_{\mathtt{b}}(w)$, resp.) for the number of occurrences of $\mathtt{a}$ ($\mathtt{b}$, resp.) in w. The* Dyck language*, written $\mathbf{D}$, is the set of words $w \in \{\mathtt{a}, \mathtt{b}\}^*$ such that (i) $\#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)$, and (ii) for every prefix v of w, $\#_{\mathtt{a}}(v) \geq \#_{\mathtt{b}}(v)$.*

The following property, which we use later, follows immediately from the definition.

Fact 1. *Let p and q be natural numbers. $\mathtt{a}^p\mathtt{b}^q\mathtt{a}^q\mathtt{b}^p \notin \mathbf{D}$ if and only if $p < q$.*

3. Main Result

This section proves the undecidability of the inclusion problem between an order-2 word language and the Dyck language (Theorem 3 below). The proof uses the following lemma, whose proof is deferred to Section 4.

Lemma 2. *Let $P_i(x_1, \ldots, x_m)(i \in \{1, 2, 3, 4\})$ be polynomials with non-negative integer coefficients. Then, one can effectively construct an order-2 HORS $\mathcal{G}$ such that*

$$\mathcal{L}(\mathcal{G}) = \{\mathtt{a}^{P_1(x_1,\ldots,x_m)}\mathtt{b}^{P_2(x_1,\ldots,x_m)}\mathtt{a}^{P_3(x_1,\ldots,x_m)}\mathtt{b}^{P_4(x_1,\ldots,x_m)} \mid x_1, \ldots, x_m \in \mathbf{Nat}\}.$$

Theorem 3. *The decision problem "Given an order-k HORS $\mathcal{G}$, does $\mathcal{L}(\mathcal{G}) \subseteq \mathbf{D}$ hold?" is undecidable for $k \geq 2$.*

[2]Note that by the definition, $\mathcal{FL}(\mathcal{G})$ does not contain ϵ.

Proof. The proof is by reduction from the undecidability of Hilbert's 10th problem (i.e., the unsolvability of Diophantine equations) [19]. Let $D(x_1, \ldots, x_m)$ be a polynomial (possibly with both positive and negative integer coefficients). Then, we have $D(x_1, \ldots, x_m) = 0$ if and only if $D(x_1, \ldots, x_m)^2 - 1 < 0$. $D(x_1, \ldots, x_m)^2 - 1$ can be represented in the form $P(x_1, \ldots, x_m) - Q(x_1, \ldots, x_m)$ by using polynomials P and Q with non-negative integer coefficients. For such polynomials P and Q, $D(x_1, \ldots, x_m) = 0$ has a solution (in natural numbers) if and only if $P(x_1, \ldots, x_m) < Q(x_1, \ldots, x_m)$ has a solution.

By Lemma 2, one can construct an order-2 HORS $\mathcal{G}$ such that:

$$\mathcal{L}(\mathcal{G}) = \{\mathtt{a}^{P(x_1,\ldots,x_m)}\mathtt{b}^{Q(x_1,\ldots,x_m)}\mathtt{a}^{Q(x_1,\ldots,x_m)}\mathtt{b}^{P(x_1,\ldots,x_m)} \mid x_1, \ldots, x_m \in \mathbf{Nat}\}.$$

By Fact 1, $\mathcal{L}(\mathcal{G}) \subseteq \mathbf{D}$ if and only if $\neg\exists x_1, \ldots, x_m \in \mathbf{Nat}.P(x_1, \ldots, x_m) < Q(x_1, \ldots, x_m)$, if and only if $\neg\exists x_1, \ldots, x_m \in \mathbf{Nat}.D(x_1, \ldots, x_m) = 0$. Since the last property is undecidable [19], so is $\mathcal{L}(\mathcal{G}) \subseteq \mathbf{D}$. $\square$

By slightly modifying the proof of the theorem above, we also obtain the following variations.

Theorem 4. *Let $L_1 = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \#_{\mathtt{a}}(w) \geq \#_{\mathtt{b}}(w)\}$ and $L_2 = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \#_{\mathtt{a}}(w) \neq \#_{\mathtt{b}}(w)\}$. The decision problems: $\mathcal{L}(\mathcal{G}) \overset{?}{\subseteq} L_1$ and $\mathcal{L}(\mathcal{G}) \overset{?}{\subseteq} L_2$ are undecidable for order-2 HORS $\mathcal{G}$.*

Proof. For $\mathcal{L}(\mathcal{G}) \overset{?}{\subseteq} L_1$, prepare the same polynomials P and Q as those in the proof of Theorem 3. By Lemma 2, one can construct an order-2 HORS $\mathcal{G}$ such that:

$$\mathcal{L}(\mathcal{G}) = \{\mathtt{a}^{P(x_1,\ldots,x_m)}\mathtt{b}^{Q(x_1,\ldots,x_m)} \mid x_1, \ldots, x_m \in \mathbf{Nat}\}$$

(let the polynomials P_1, P_2, P_3 and P_4 be P, Q, 0 and 0 respectively). Then, $D(x_1, \ldots, x_m) = 0$ is not satisfiable if and only if $P(x_1, \ldots, x_m) \geq Q(x_1, \ldots, x_m)$ for all $x_1, \ldots, x_m \in \mathbf{Nat}$, if and only if $\mathcal{L}(\mathcal{G}) \subseteq L_1$. For $\mathcal{L}(\mathcal{G}) \overset{?}{\subseteq} L_2$, let $P(x_1, \ldots, x_m)$ and $Q(x_1, \ldots, x_m)$ be polynomials with non-negative coefficients such that $D(x_1, \ldots, x_m) = P(x_1, \ldots, x_m) - Q(x_1, \ldots, x_m)$, and construct an order-2 HORS $\mathcal{G}$ such that:

$$\mathcal{L}(\mathcal{G}) = \{\mathtt{a}^{P(x_1,\ldots,x_m)}\mathtt{b}^{Q(x_1,\ldots,x_m)} \mid x_1, \ldots, x_m \in \mathbf{Nat}\}.$$

Then, $D(x_1, \ldots, x_m) = 0$ is not satisfiable if and only if $\mathcal{L}(\mathcal{G}) \subseteq L_2$. $\square$

Remark 1. *We do not know whether the following problem is decidable for order-2 HORS $\mathcal{G}$.*

$$\mathcal{L}(\mathcal{G}) \overset{?}{\subseteq} \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid \#_{\mathtt{a}}(w) = \#_{\mathtt{b}}(w)\}.$$

4. Proof of Lemma 2

To clarify the idea of the construction of a HORS that satisfies the property of Lemma 2, we first give an order-3 HORS that generates the same word language:

$$L_0 = \{\mathtt{a}^{P_1(x_1,\ldots,x_m)}\mathtt{b}^{P_2(x_1,\ldots,x_m)}\mathtt{a}^{P_3(x_1,\ldots,x_m)}\mathtt{b}^{P_4(x_1,\ldots,x_m)} \mid x_1,\ldots,x_m \in \mathbf{Nat}\}.$$

We then modify the construction to obtain an order-2 HORS that generates the same language.

The idea of the first part is to construct a HORS that corresponds to the following grammar, which uses natural numbers and operations on them in addition to ordinary primitives for HORS.

$$\begin{array}{l}
S \to F\,0\,\cdots\,0.\\
F\,x_1\,\cdots\,x_m \to G\,x_1\,\cdots\,x_m.\\
F\,x_1\,\cdots\,x_m \to F\,(x_1+1)\,x_2\,\cdots\,x_m.\\
\cdots\\
F\,x_1\,\cdots\,x_m \to F\,x_1\,x_2\,\cdots\,(x_m+1).\\
G\,x_1\,\cdots\,x_m \to A_{P_1}\,x_1\,\cdots\,x_m(B_{P_2}\,x_1\,\cdots\,x_m(A_{P_3}\,x_1\,\cdots\,x_m(B_{P_4}\,x_1\,\cdots\,x_m\,\mathtt{e}))).\\
A_{P_1}\,x_1\,\cdots\,x_m\,y \to \mathtt{a}^{P_1(x_1,\ldots,x_m)}(y).\\
B_{P_2}\,x_1\,\cdots\,x_m\,y \to \mathtt{b}^{P_2(x_1,\ldots,x_m)}(y).\\
A_{P_3}\,x_1\,\cdots\,x_m\,y \to \mathtt{a}^{P_3(x_1,\ldots,x_m)}(y).\\
B_{P_4}\,x_1\,\cdots\,x_m\,y \to \mathtt{b}^{P_4(x_1,\ldots,x_m)}(y).
\end{array}$$

By using the rules for S and F, we can rewrite S to $G\,n_1\,\cdots\,n_m$ for any natural numbers $n_1,\ldots,n_m$. The term $G\,n_1\,\cdots\,n_m$ can then be rewritten to:

$$\mathtt{a}^{P_1(n_1,\ldots,n_m)}(\mathtt{b}^{P_2(n_1,\ldots,n_m)}(\mathtt{a}^{P_3(n_1,\ldots,n_m)}(\mathtt{b}^{P_4(n_1,\ldots,n_m)}\mathtt{e}))),$$

by using the rules for G, A_{P_1}, B_{P_2}, A_{P_3}, and B_{P_4}. Thus, the above rewriting rules generate the language L_0.

It remains to remove operations on natural numbers, which can be easily performed in the order-3 case, by using the Church encoding:

$$\begin{array}{ll}
\mathit{Zero}\,x\,y \to y. & \mathit{Succ}\,n\,x\,y \to n\,x\,(x\,y).\\
\mathit{Plus}\,m\,n\,x\,y \to m\,x\,(n\,x\,y). & \mathit{Mult}\,m\,n\,x\,y \to m\,(n\,x)y.
\end{array}$$

Here, non-terminals have the following types:

$$\begin{array}{l}
\mathit{Zero} : \mathtt{nat}\\
\mathit{Succ} : \mathtt{nat} \to \mathtt{nat}\\
\mathit{Plus}, \mathit{Mult} : \mathtt{nat} \to \mathtt{nat} \to \mathtt{nat}
\end{array}$$

where $\mathtt{nat}$ is an abbreviation of $(\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \to \mathtt{o}$. (Note that since HORS must be simply-typed, not all operations on natural numbers can be represented as non-terminals; for example, we cannot express subtraction.) By using the encodings above, we can replace 0 and $x_i + 1$ with *Zero* and *Succ* x_i respectively. We can also define non-terminals P'_i ($i \in \{1,2,3,4\}$) such that $P'_i\ x_1 \cdots x_m\, y\, z, \ldots,$ reduces to $y^{P_i(x_1,\ldots,x_m)}(z)$. For example, if $m = 2$ and $P_i(x_1, x_2) = x_1^2 + x_2$, then P'_i can be defined by:

$$P'_i\ x_1\ x_2\ y\ z \to \mathit{Plus}\ (\mathit{Mult}\, x_1\ x_1)\ x_2\ y\ z.$$

Thus, the rules for $A_{P_1}, B_{P_2}, A_{P_3}, B_{P_4}$ can be replaced with:

$$\begin{array}{l} A_{P_1}\, x_1\ \cdots\ x_m\, z \to P'_1\, x_1\ \cdots\ x_m\, \mathtt{a}\, y. \\ B_{P_2}\, x_1\ \cdots\ x_m\, z \to P'_2\, x_1\ \cdots\ x_m\, \mathtt{b}\, y. \\ A_{P_3}\, x_1\ \cdots\ x_m\, z \to P'_3\, x_1\ \cdots\ x_m\, \mathtt{a}\, y. \\ B_{P_4}\, x_1\ \cdots\ x_m\, z \to P'_4\, x_1\ \cdots\ x_m\, \mathtt{b}\, y. \end{array}$$

The resulting rewriting rules form an order-3 HORS, which generates the language L_0.

We now modify the above construction to obtain an order-2 HORS that generates the same language. The idea is, instead of passing the values of $x_1, \ldots, x_m$ (of type $\mathtt{nat} = (\mathtt{o} \to \mathtt{o}) \to \mathtt{o} \to \mathtt{o}$) as parameters of F, to pass $\lambda y.\mathtt{a}^{x_1^{k_1} \cdots x_m^{k_m}}(y)$ (abbreviated to $\mathtt{a}^{x_1^{k_1} \cdots x_m^{k_m}}$ below) and $\lambda y.\mathtt{b}^{x_1^{k_1} \cdots x_m^{k_m}}(y)$, of type $\mathtt{o} \to \mathtt{o}$. For example, if $m = 1$, $P_1(x) = P_4(x) = x^2 + 2$ and $P_2(x) = P_3(x) = 2x^2 + 1$, then we pass $\mathtt{a}^{x^2}$, $\mathtt{a}^{x^1} (= \mathtt{a}^x)$, $\mathtt{a}^{x^0} (= \mathtt{a})$, $\mathtt{b}^{x^2}$, $\mathtt{b}^{x^1}$ and $\mathtt{b}^{x^0}$ to F for each natural number x. Thus, in this case, an order-2 HORS that generates L_0 is given by:

$$\begin{array}{l} S \to F\, I\, I\, \mathtt{a}\, I\, I\, \mathtt{b}. \\ F\, v_{x^2}\, v_x\, v_1\, w_{x^2}\, w_x\, w_1 \to G\, v_{x^2}\, v_x\, v_1\, w_{x^2}\, w_x\, w_1. \\ F\, v_{x^2}\, v_x\, v_1\, w_{x^2}\, w_x\, w_1 \to \\ \qquad F\, (A_{(x+1)^2}\, v_{x^2}\, v_x\, v_1)\, (A_{x+1}\, v_x\, v_1)\, v_1\, (A_{(x+1)^2}\, w_{x^2}\, w_x\, w_1)\, (A_{x+1}\, v_x\, w_1)\, w_1. \\ I\, y \to y. \\ A_{(x+1)^2}\, z_{x^2}\ z_x\ z_1\, y \to z_{x^2}(z_x(z_x\, (z_1\, y))). \\ A_{x+1}\, z_x\, z_1\, y \to z_x\, (z_1\, y). \\ G\, v_{x^2}\, v_x\, v_1\, w_{x^2}\, w_x\, w_1 \to P_1\, v_{x^2}\, v_x\, v_1\, (P_2\, w_{x^2}\, w_x\, w_1)(P_2\, v_{x^2}\, v_x\, v_1\, (P_1\, w_{x^2}\, w_x\, w_1\, \mathtt{e})). \\ P_1\, z_{x^2}\, z_x\, z_1\, y \to z_{x^2}\, (z_1\, (z_1\, y)). \\ P_2\, z_{x^2}\, z_x\, z_1\, y \to z_{x^2}\, (z_{x^2}\, (z_1 y)). \end{array}$$

The formal parameters $v_{x^2}, v_x, v_1, w_{x^2}, w_x, w_1$ of F are bound to $\mathtt{a}^{x^2}$, $\mathtt{a}^x$, $\mathtt{a}^1$, $\mathtt{b}^{x^2}, \mathtt{b}^x$, and $\mathtt{b}^1$ respectively. The non-terminal $A_{(x+1)^2}$ defined in the fifth rule computes $a^{(x+1)^2}$, given a^{x^2}, a^x, and a as the values of z_{x^2}, z_x and z_1, using the fact $a^{(x+1)^2}(y) = a^{x^2+x+x+1}(y) = a^{x^2}(a^x(a^x(a\,y)))$. Similarly, A_{x+1} computes a^{x+1}, given a^x and a as values of z_x and z_1. Thus, for every natural number n, S can be rewritten to $G\ \mathtt{a}^{n^2}\ \mathtt{a}^n\ \mathtt{a}^1\ \mathtt{b}^{n^2}\ \mathtt{b}^n\ \mathtt{b}^1$ (modulo the semantic equivalence of terms). The non-terminals P_1 and P_2 defined in the last two rules compute $a^{P_1(n)}$ and $a^{P_2(n)}$ respectively, given a^{n^2}, a^n, and a as the values of z_{n^2}, z_n and z_1; thus $G\ \mathtt{a}^{n^2}\ \mathtt{a}^n\ \mathtt{a}^1\ \mathtt{b}^{n^2}\ \mathtt{b}^n\ \mathtt{b}^1$ can be rewritten to

$$\mathtt{a}^{P_1(n)}(\mathtt{b}^{P_2(n)}(\mathtt{a}^{P_3(n)}(\mathtt{b}^{P_4(n)}\mathtt{e}))),$$

for $P_1(x) = P_4(x) = x^2 + 2$ and $P_2(x) = P_3(x) = 2x^2 + 1$.

We now present the general construction. Let $d_i (i \in \{1, \ldots, m\})$ be the largest degree of $P_1(x_1, \ldots, x_m) + \cdots + P_4(x_1, \ldots, x_m)$ in variable x_i. Then, each $P_i(x_1, \ldots, x_m)$ can be represented as a linear combination of monomials $x_1^{j_1} \cdots x_m^{j_m}$ $(j_1 \leq d_1, \ldots, j_m \leq d_m)$:

$$\sum_{j_1 \leq d_1, \ldots, j_m \leq d_m} c_{i,j_1,\ldots,j_m} x_1^{j_1} \cdots x_m^{j_m}.$$

Let ℓ be $(d_1 + 1) \cdots (d_m + 1)$. Then, an order-2 HORS $\mathcal{G}_{L_0}$ that generates L_0 is given in Figure 1. Here, $\widetilde{v}$ and $\widetilde{w}$ denote the sequences of variables $v_{x_1^{d_1} \cdots x_m^{d_m}} \cdots v_{x_1^0 \cdots x_m^0}$ and $w_{x_1^{d_1} \cdots x_m^{d_m}} \cdots w_{x_1^0 \cdots x_m^0}$ respectively. The non-terminal $A_{x_1^{j_1} \cdots (x_k+1)^{j_k} \cdots x_m^{j_m}}$ computes $a^{n_1^{j_1} \cdots (n_k+1)^{j_k} \cdots n_m^{j_m}}$, given $a^{n_1^{d_1} \cdots n_m^{d_m}}, \ldots,$ $a^{n_1^0 \cdots n_m^0}$ as the values of $\widetilde{v} = v_{x_1^{d_1} \cdots x_m^{d_m}}, \ldots, v_{x_1^0 \cdots x_m^0}$. Thus, for every tuple $(n_1, \ldots, n_m)$ of natural numbers, S can be rewritten to (modulo the semantic equivalence of terms):

$$G\ \mathtt{a}^{n_1^{d_1} \cdots n_m^{d_m}}\ \cdots\ \mathtt{a}^{n_1^0 \cdots n_m^0}\ \mathtt{b}^{n_1^{d_1} \cdots n_m^{d_m}}\ \cdots\ \mathtt{b}^{n_1^0 \cdots n_m^0}.$$

It can then be rewritten to

$$\mathtt{a}^{P_1(n_1,\ldots,n_m)}(\mathtt{b}^{P_2(n_1,\ldots,n_m)}(\mathtt{a}^{P_3(n_1,\ldots,n_m)}(\mathtt{b}^{P_4(n_1,\ldots,n_m)}\mathtt{e}))),$$

by using the rules for G and P_i. From the discussion above, it should be clear that $\mathcal{L}(\mathcal{G}_{L_0}) = L_0$. This completes the proof.

5. Related Work

Arnold and Nivat [3] considered non-deterministic recursive program schemes (i.e. order-1 non-deterministic higher-order recursion schemes) as

$$\mathcal{G}_{L_0} = (\{\mathtt{a} \mapsto 1, \mathtt{b} \mapsto 1\}, \mathcal{N}, \mathcal{R}, S)$$

$$\begin{aligned}\mathcal{N} = \{&S \mapsto \mathtt{o}, F \mapsto (\mathtt{o} \to \mathtt{o})^{2\ell} \to \mathtt{o}, G \mapsto (\mathtt{o} \to \mathtt{o})^{2\ell} \to \mathtt{o}, I \mapsto \mathtt{o} \to \mathtt{o}\} \\ &\cup \{A_{x_1^{j_1} \cdots x_m^{j_m}} \mapsto (\mathtt{o} \to \mathtt{o})^{\ell} \to \mathtt{o} \to \mathtt{o} \mid i \in \{1, \ldots, m\}, j_i \in \{0, \ldots, d_i\}\} \\ &\cup \{P_i \mapsto (\mathtt{o} \to \mathtt{o})^{\ell} \to \mathtt{o} \to \mathtt{o} \mid i \in \{1,2,3,4\}\}\end{aligned}$$

$\mathcal{R}$ consists of:

$S \to F \underbrace{I \cdots I}_{\ell-1} \mathtt{a} \underbrace{I \cdots I}_{\ell-1} \mathtt{b}.$

$F\,\widetilde{v}\,\widetilde{w} \to G\,\widetilde{v}\,\widetilde{w}.$

$F\,\widetilde{v}\,\widetilde{w} \to$

$\quad F\,(A_{(x_1+1)^{d_1} \cdots x_m^{d_m}}\,\widetilde{v}) \cdots (A_{(x_1+1)^0 \cdots x_m^0}\,\widetilde{v})\,(A_{(x_1+1)^{d_1} \cdots x_m^{d_m}}\,\widetilde{w}) \cdots (A_{(x_1+1)^0 \cdots x_m^0}\,\widetilde{w}).$

$\cdots$

$F\,\widetilde{v}\,\widetilde{w} \to$

$\quad F\,(A_{x_1^{d_1} \cdots (x_m+1)^{d_m}}\,\widetilde{v}) \cdots (A_{x_1^0 \cdots (x_m+1)^0}\,\widetilde{v})\,(A_{x_1^{d_1} \cdots (x_m+1)^{d_m}}\,\widetilde{w}) \cdots (A_{x_1^0 \cdots (x_m+1)^0}\,\widetilde{w}).$

$I\,y \to y.$

$A_{x_1^{j_1} \cdots (x_k+1)^{j_k} \cdots x_m^{j_m}}\,\widetilde{v}\,y \to$

$\quad v_{x_1^{i_1} \cdots x_k^{j_k} \cdots x_m^{j_m}}(\cdots (v_{x_1^{i_1} \cdots x_k^{j} \cdots x_m^{j_m}})^{\binom{j_k}{j}}(\cdots (v_{x_1^{i_1} \cdots x_k^{0} \cdots x_m^{j_m}}\,y) \cdots) \cdots)$

(for each $k \in \{1, \ldots, m\}, j_1 \le d_1, \ldots, j_m \le d_m$).

$G\,\widetilde{v}\,\widetilde{w} \to P_1\,\widetilde{v}\,(P_2\,\widetilde{w}(P_3\,\widetilde{v}(P_4\,\widetilde{w}\,\mathtt{e}))).$

$P_i\,\widetilde{v}\,y \to (v_{x_1^{d_1} \cdots x_m^{d_m}})^{c_{i,d_1,\ldots,d_m}}(\cdots ((v_{x_1^0 \cdots x_m^0})^{c_{i,0,\ldots,0}}\,y) \cdots)$ (for each $i \in \{1,2,3,4\}$).

Figure 1: An order-2 HORS $\mathcal{G}_{L_0}$ that generates L_0.

generators of both finite tree languages ($\mathcal{TL}(\mathcal{G})$ in Section 2) and infinite tree languages, and gave fixpoint characterizations.

Higher-order grammars as generators of word/tree languages have been actively studied, especially in 1970's and 80's (see, e.g. [6, 23]). To our knowledge, however, the inclusion problem with the Dyck language has not been studied until recently. As already mentioned, Uezato and Minamide [22] have shown that the inclusion between an indexed language and the Dyck language is undecidable. Since the class of indexed languages and the class of order-2 word languages (which is also known as OI languages) coincide and there exist effective mutual translations ([21], Theorem 5.3), the result of Uezato and Minamide [22] is the same as ours. Their proof is somewhat indirect, however, which uses an undecidability result on DT0L systems [27]. Our proof is arguably more direct, demonstrating the power of order-2 grammars to express polynomials.

6. Conclusion

We have shown that the inclusion between an order-2 word language and the Dyck language is undecidable, by a reduction from the undecidability of Hilbert's 10th problem. By using the technique of [10, 9] to convert higher-order functional programs to HORS that generate the set of all the event sequences, one can deduce from the undecidability result that certain program verification problems like "Does a program output a string conforming to a valid HTML?" and "Does a program access a certain resource in a stack-like manner?" are undecidable even for a restricted, Turing-incomplete higher-order language (more precisely, for the simply-typed λ-calculus with recursion and primitives for printing strings or accessing resources).

Acknowledgment.. We thank anonymous referees for useful comments. This work was supported by JSPS KAKENHI Grant Number JP15H05706.

[1] M. Nivat, Langages algébriques sur le magma libre et sémantique des schémas de programme, in: ICALP, 1972, pp. 293–308.

[2] M. Nivat, On the interpretation of recursive program schemes, in: Symposia Mathematica, 1975, pp. 255–281.

[3] A. Arnold, M. Nivat, Non deterministic recursive program schemes, in: FCT, 1977, pp. 12–21.

[4] B. Courcelle, M. Nivat, The algebraic semantics of recursive program schemes, in: Proceedings of MFCS 1978, Vol. 64 of Lecture Notes in Computer Science, 1978, pp. 16–30.

[5] W. Damm, Higher type program schemes and their tree languages, in: Theoretical Computer Science, 3rd GI-Conference, Vol. 48 of Lecture Notes in Computer Science, 1977, pp. 51–72.

[6] W. Damm, The IO- and OI-hierarchies, Theoretical Computer Science 20 (1982) 95–207.

[7] T. Knapik, D. Niwinski, P. Urzyczyn, Higher-order pushdown trees are easy, in: FoSSaCS 2002, Vol. 2303 of Lecture Notes in Computer Science, Springer, 2002, pp. 205–222.

[8] C.-H. L. Ong, On model-checking trees generated by higher-order recursion schemes, in: LICS 2006, IEEE Computer Society Press, 2006, pp. 81–90.

[9] N. Kobayashi, Model checking higher-order programs, Journal of the ACM 60 (3).

[10] N. Kobayashi, Types and higher-order recursion schemes for verification of higher-order programs, in: Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages, ACM Press, 2009, pp. 416–428.

[11] N. Kobayashi, R. Sato, H. Unno, Predicate abstraction and CEGAR for higher-order model checking, in: Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, ACM Press, 2011, pp. 222–233.

[12] C.-H. L. Ong, S. Ramsay, Verifying higher-order programs with pattern-matching algebraic data types, in: Proceedings of ACM SIGPLAN/SIGACT Symposium on Principles of Programming Languages, ACM Press, 2011, pp. 587–598.

[13] O. Kupferman, M. Y. Vardi, P. Wolper, An automata-theoretic approach to branching-time model checking, Journal of the Association for Computing Machinery (JACM) 47 (2) (2000) 312–360.

[14] M. Y. Vardi, An automata-theoretic approach to linear temporal logic, in: F. Moller, G. M. Birtwistle (Eds.), Logics for Concurrency - Structure versus Automata (8th Banff Higher Order Workshop, August 27 -

September 3, 1995, Proceedings), Vol. 1043 of Lecture Notes in Computer Science, Springer, 1995, pp. 238–266.

[15] J. E. Hopcroft, J. D. Ullman, Introduction to Automata Theory, Languages, and Computation, Addison-Wesley, 1979.

[16] S. A. Greibach, E. P. Friedman, Superdeterministic PDAs: A subcase with a decidable inclusion problem, J. ACM 27 (4) (1980) 675–700.

[17] A. Tozawa, Y. Minamide, Complexity results on balanced context-free languages, in: H. Seidl (Ed.), Foundations of Software Science and Computational Structures, 10th International Conference, FOSSACS 2007, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2007, Braga, Portugal, March 24-April 1, 2007, Proceedings, Vol. 4423 of Lecture Notes in Computer Science, Springer, 2007, pp. 346–360.

[18] A. Bertoni, C. Choffrut, R. Radicioni, The inclusion problem of context-free languages: Some tractable cases, Int. J. Found. Comput. Sci. 22 (2) (2011) 289–299.

[19] Y. V. Matiyasevich, Hilbert's Tenth Problem, The MIT Press, 1993.

[20] M. Nivat, On some families of languages related to the dyck language, in: P. C. Fischer, R. Fabian, J. D. Ullman, R. M. Karp (Eds.), Proceedings of the 2nd Annual ACM Symposium on Theory of Computing, May 4-6, 1970, Northampton, Massachusetts, USA, ACM, 1970, pp. 221–225.

[21] M. J. Fischer, Grammars with macro-like productions, in: 9th Annual Symposium on Switching and Automata Theory, Schenectady, New York, USA, October 15-18, 1968, IEEE Computer Society, 1968, pp. 131–142.

[22] Y. Uezato, Y. Minamide, Monoid-based approach to the inclusion problem on superdeterministic pushdown automata, in: S. Brlek, C. Reutenauer (Eds.), Developments in Language Theory - 20th International Conference, DLT 2016, Montréal, Canada, July 25-28, 2016, Proceedings, Vol. 9840 of Lecture Notes in Computer Science, Springer, 2016, pp. 393–405.

[23] G. M. Kobele, S. Salvati, The IO and OI hierarchies revisited, Inf. Comput. 243 (2015) 205–221.

[24] K. Aehlig, J. G. de Miranda, C.-H. L. Ong, Safety is not a restriction at level 2 for string languages, in: FoSSaCS, Vol. 3441 of Lecture Notes in Computer Science, Springer, 2005, pp. 490–504.

[25] M. Wand, An algebraic formulation of the Chomsky hierarchy, in: Category Theory Applied to Computation and Control, Vol. 25 of Lecture Notes in Computer Science, Springer, 1974, pp. 209–213.

[26] K. Asada, N. Kobayashi, On Word and Frontier Languages of Unsafe Higher-Order Grammars, in: 43rd International Colloquium on Automata, Languages, and Programming (ICALP 2016), Vol. 55 of LIPIcs, Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016, pp. 111:1–111:13.

[27] A. Salomaa, M. Soittola, Automata-theoretic Aspects of Formal Power Series, Springer-Verlag, 1977.