Pumping Lemma for Higher-Order Languages*†

Kazuyuki Asada[1] and Naoki Kobayashi[2]

1 **The University of Tokyo, Tokyo, Japan**
asada@kb.is.s.u-tokyo.ac.jp

2 **The University of Tokyo, Tokyo, Japan**
koba@is.s.u-tokyo.ac.jp

Abstract

We study a pumping lemma for the word/tree languages generated by higher-order grammars. Pumping lemmas are known up to order-2 word languages (i.e., for regular/context-free/indexed languages), and have been used to show that a given language does not belong to the classes of regular/context-free/indexed languages. We prove a pumping lemma for word/tree languages of arbitrary orders, modulo a conjecture that a higher-order version of Kruskal's tree theorem holds. We also show that the conjecture indeed holds for the order-2 case, which yields a pumping lemma for order-2 tree languages and order-3 word languages.

1998 ACM Subject Classification F.4.3 Formal Languages

Keywords and phrases pumping lemma, higher-order grammars, Kruskal's tree theorem

Digital Object Identifier 10.4230/LIPIcs.ICALP.2017.97

1 Introduction

We study a pumping lemma for higher-order languages, i.e., the languages generated by higher-order word/tree grammars where non-terminals can take higher-order functions as parameters. The classes of higher-order languages [26, 18, 4, 5, 6] form an infinite hierarchy, where the classes of order-0, order-1, and order-2 languages are those of regular, context-free and indexed languages. Higher-order grammars and languages have been extensively studied by Damm [4] and Engelfriet [5, 6] and recently re-investigated in the context of model checking and program verification [9, 20, 15, 24, 11, 16, 12, 23].

Pumping lemmas [2, 7] are known up to order-2 word languages, and have been used to show that a given language does not belong to the classes of regular/context-free/indexed languages. To our knowledge, however, little is known about languages of order-3 or higher. Pumping lemmas [21, 12] are also known for higher-order *deterministic* grammars (as generators of infinite *trees*, rather than tree languages), but they cannot be applied to non-deterministic grammars.

In the present paper, we state and prove a pumping lemma for unsafe[1] languages of arbitrary orders modulo an assumption that a "higher-order version" of Kruskal's tree theorem [17, 19] holds. Let $\preceq$ be the homeomorphic embedding on finite ranked trees[2], and

* A full version of the paper is available at http://arxiv.org/abs/1705.10699.
† This work was supported by JSPS Kakenhi 15H05706.

[1] See, e.g., [16] for the distinction between safe vs unsafe languages; the class of unsafe languages subsumes that of safe languages.

[2] I.e., $T_1 \preceq T_2$ if there exists an injective map from the nodes of T_1 to those of T_2 that preserves the labels of nodes and the ancestor/descendant-relation of nodes; see Section 2 for the precise definition.

44th International Colloquium on Automata, Languages, and Programming (ICALP 2017).
Editors: Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl;
Article No. 97; pp. 97:1–97:14
Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

$\prec$ be the strict version of $\preceq$. The statement of our pumping lemma[3] is that for any order-n infinite tree language L, there exist a constant c and a strictly increasing infinite sequence of trees $T_0 \prec T_1 \prec T_2 \prec \cdots$ in L such that $|T_i| \leq \mathbf{exp}_n(ci)$ for every $i \geq 0$, where $\mathbf{exp}_0(x) = x$ and $\mathbf{exp}_{n+1}(x) = 2^{\mathbf{exp}_n(x)}$. Due to the correspondence between word/tree languages [4, 1], it also implies that for any order-n infinite word language L (where $n \geq 1$), there exist a constant c and a strictly increasing infinite sequence of words $w_0 \prec w_1 \prec w_2 \prec \cdots$ in L such that $|w_i| \leq \mathbf{exp}_{n-1}(ci)$ for every $i \geq 0$, where $\prec$ is the subsequence relation. The pumping lemma can be used, for example, to show (modulo the conjecture) that the order-$(n+1)$ language $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$ does not belong to the class of order-n word languages, for $n > 0$. Thus the lemma would also provide an alternative proof of the strictness of the hierarchy of the classes of higher-order languages.[4]

We now informally explain the assumption of "higher-order Kruskal's tree theorem" (see Section 2 for details). Kruskal's tree theorem [17, 19] states that the homeomorphic embedding $\preceq$ is a well-quasi order, i.e., that for any infinite sequence of trees $T_0, T_1, T_2, \ldots$, there exist $i < j$ such that $T_i \preceq T_j$. The homeomorphic embedding $\preceq$ can be naturally lifted (e.g. via the logical relation) to a family of relations $(\preceq_\kappa)_\kappa$ on higher-order tree functions of type κ. Our conjecture of "higher-order Kruskal's theorem" states that, for every simple type κ, $\preceq_\kappa$ is also a well-quasi order on the functions expressed by the simply-typed λ-terms. We prove that the conjecture indeed holds up to order-2 functions, if we take $\preceq_\kappa$ as the logical relation induced from the homeomorphic embedding $\preceq$. Thus, our pumping "lemma" is indeed true for order-2 tree languages and order-3 word languages. To our knowledge, the pumping lemma for those languages is novel. The conjecture remains open for order-3 or higher, which should be of independent interest.

Our proof of the pumping lemma (modulo the conjecture) uses the recent work of Parys [23] on an intersection type system for deciding the infiniteness of the language generated by a given higher-order grammar, and our previous work on the relationship between higher-order word/tree languages [1].

The rest of this paper is organized as follows. Section 2 prepares several definitions and states our pumping lemma and the conjecture more formally. Section 3 derives some corollaries of Parys' result [23]. Section 4 prepares a simplified and specialized version of our previous result [1]. Using the results in Sections 3 and 4, we prove our pumping lemma (modulo the conjecture) in Section 5. Section 6 proves the conjecture on higher-order Kruskal's tree theorem for the order-2 case, by which we obtain the (unconditional) pumping lemma for order-2 tree languages and order-3 word languages. Section 7 discusses related work and Section 8 concludes.

2 Preliminaries

We first give basic definitions needed for explaining our main theorem. We then state the main theorem and provide an overview of its proof.

---

[3] This should perhaps be called a pumping "conjecture" since it relies on the conjecture of the higher-order Kruskal's tree theorem.

[4] The strictness of the hierarchy of higher-order *safe* languages has been shown by Engelfriet [5] using a complexity argument, and Kartzow [8] observed that essentially the same argument is applicable to obtain the strictness of the hierarchy of unsafe languages as well. Their argument cannot be used for showing that a particular language does not belong to the class of order-n languages.

2.1 λ-terms and Higher-order Grammars

This section gives basic definitions for terms and higher-order grammars.

▶ **Definition 1** (Types and Terms). The set of *simple types*, ranged over by κ, is given by: $\kappa ::= \mathtt{o} \mid \kappa_1 \to \kappa_2$. The order of a simple type κ, written $\mathtt{order}(\kappa)$ is defined by $\mathtt{order}(\mathtt{o}) = 0$ and $\mathtt{order}(\kappa_1 \to \kappa_2) = \max(\mathtt{order}(\kappa_1) + 1, \mathtt{order}(\kappa_2))$. The type $\mathtt{o}$ describes trees, and $\kappa_1 \to \kappa_2$ describes functions from κ_1 to κ_2. The set of $\lambda^{\to,+}$*-terms* (or *terms*), ranged over by s, t, u, v, is defined by:

$$t ::= x \mid a\, t_1 \cdots t_k \mid t_1\, t_2 \mid \lambda x : \kappa.t \mid t_1 + t_2\,.$$

Here, x ranges over variables, and a over constants (which represent tree constructors). Variables are also called *non-terminals*, ranged over by x, y, z, f, g, A, B; and constants are also called *terminals*. A ranked alphabet Σ is a map from a finite set of terminals to natural numbers called *arities*; we implicitly assume a ranked alphabet whose domain contains all terminals discussed, unless explicitly described. $+$ is non-deterministic choice. As seen below, our simple type system forces that a terminal must be fully applied; this does not restrict the expressive power, as $\lambda x_1, \dots, x_k.a\, x_1 \cdots x_k$ is available. We often omit the type κ of $\lambda x : \kappa.t$. A term is called an *applicative term* if it does not contain λ-abstractions nor $+$, and called a $\lambda^{\to}$*-term* if it does not contain $+$. As usual, we identify terms up to the α-equivalence, and implicitly apply α-conversions.

A (simple) type environment $\mathcal{K}$ is a sequence of type bindings of the form $x : \kappa$ such that if $\mathcal{K}$ contains $x : \kappa$ and $x' : \kappa'$ in different positions then $x \neq x'$. In type environments, non-terminals are also treated as variables. A term t has type κ under $\mathcal{K}$ if $\mathcal{K} \vdash_{\mathtt{ST}} t : \kappa$ is derivable from the following typing rules.

$$\frac{}{\mathcal{K},\, x : \kappa,\, \mathcal{K}' \vdash_{\mathtt{ST}} x : \kappa} \qquad \frac{\Sigma(a) = k \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_i : \mathtt{o} \quad (\text{for each } i \in \{1, \dots, k\})}{\mathcal{K} \vdash_{\mathtt{ST}} a\, t_1 \cdots t_k : \mathtt{o}}$$

$$\frac{\mathcal{K} \vdash_{\mathtt{ST}} t_1 : \kappa_2 \to \kappa \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_2 : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} t_1\, t_2 : \kappa} \qquad \frac{\mathcal{K},\, x : \kappa_1 \vdash_{\mathtt{ST}} t : \kappa_2}{\mathcal{K} \vdash_{\mathtt{ST}} \lambda x : \kappa_1.t : \kappa_1 \to \kappa_2} \qquad \frac{\mathcal{K} \vdash_{\mathtt{ST}} t_1 : \mathtt{o} \qquad \mathcal{K} \vdash_{\mathtt{ST}} t_2 : \mathtt{o}}{\mathcal{K} \vdash_{\mathtt{ST}} t_1 + t_2 : \mathtt{o}}$$

We consider below only well-typed terms. Note that given $\mathcal{K}$ and t, there exists at most one type κ such that $\mathcal{K} \vdash_{\mathtt{ST}} t : \kappa$. We call κ the type of t (with respect to $\mathcal{K}$). We often omit "with respect to $\mathcal{K}$" if $\mathcal{K}$ is clear from context. The (internal) *order* of t, written $\mathtt{order}_{\mathcal{K}}(t)$, is the largest order of the types of subterms of t, and the *external order* of t, written $\mathtt{eorder}_{\mathcal{K}}(t)$, is the order of the type of t (both with respect to $\mathcal{K}$). We often omit $\mathcal{K}$ when it is clear from context. For example, for $t = (\lambda x : \mathtt{o}.x)\mathtt{e}$, $\mathtt{order}_{\emptyset}(t) = 1$ and $\mathtt{eorder}_{\emptyset}(t) = 0$.

We call a term t *ground* (with respect to $\mathcal{K}$) if $\mathcal{K} \vdash_{\mathtt{ST}} t : \mathtt{o}$. We call t a (finite, Σ-ranked) *tree* if t is a closed ground applicative term consisting of only terminals. We write $\mathbf{Tree}_{\Sigma}$ for the set of Σ-ranked trees, and use the meta-variable π for trees.

The set of *contexts*, ranged over by C, D, G, H, is defined by $C ::= [\,] \mid C\, t \mid t\, C \mid \lambda x.C$. We write $C[t]$ for the term obtained from C by replacing $[\,]$ with t. Note that the replacement may capture variables; e.g., $(\lambda x.[\,])[x]$ is $\lambda x.x$. We call C a $(\mathcal{K}', \kappa')$-$(\mathcal{K}, \kappa)$*-context* if $\mathcal{K} \vdash_{\mathtt{ST}} C : \kappa$ is derived by using axiom $\mathcal{K}' \vdash_{\mathtt{ST}} [\,] : \kappa'$. We also call a $(\emptyset, \kappa')$-$(\emptyset, \kappa)$-context a κ'-κ*-context*. The (internal) *order* of a $(\mathcal{K}', \kappa')$-$(\mathcal{K}, \kappa)$-context, is the largest order of the types occurring in the derivation of $\mathcal{K} \vdash_{\mathtt{ST}} C : \kappa$. A context is called a $\lambda^{\to}$*-context* if it does not contain $+$.

We define the *size* $|t|$ of a term t by: $|x| := 1$, $|a\, t_1 \cdots, t_k| := 1 + |t_1| + \cdots + |t_k|$, $|s\, t| := |s| + |t| + 1$, $|\lambda x.t| := |t| + 1$, and $|s + t| := |s| + |t| + 1$. The size $|C|$ of a context C is defined similarly, with $|[\,]| := 0$.

▶ **Definition 2** (Reduction and Language)**.** The set of *(call-by-name) evaluation contexts* is defined by:

$$E ::= [\,]\, t_1 \cdots t_k \mid a\, \pi_1 \cdots \pi_i\, E\, t_1 \cdots t_k$$

and the *call-by-name reduction* for (possibly open) ground terms is defined by:

$$E[(\lambda x.t)t'] \longrightarrow E[[t'/x]t] \qquad\qquad E[t_1 + t_2] \longrightarrow E[t_i] \quad (i = 1, 2)$$

where $[t'/x]t$ is the usual capture-avoiding substitution. We write $\longrightarrow^*$ for the reflexive transitive closure of $\longrightarrow$. A *call-by-name normal form* is a ground term t such that $t \not\longrightarrow t'$ for any t'. For a closed ground term t, we define the *tree language $\mathcal{L}(t)$ generated by t* by $\mathcal{L}(t) := \{\pi \mid t \longrightarrow^* \pi\}$. For a closed ground $\lambda^{\to}$-term t, $\mathcal{L}(t)$ is a singleton set $\{\pi\}$; we write $\mathcal{T}(t)$ for such π and call it *the tree of t*.

Note that $t \longrightarrow^* t'$ implies $[s/x]t \longrightarrow^* [s/x]t'$, and that the set of call-by-name normal forms equals the set of trees and ground terms of the form $E[x]$.

For $x : \kappa \vdash_{\mathsf{ST}} t : \mathsf{o}$ where t does not contain the non-deterministic choice, t is called *linear* (with respect to x) if x occurs exactly once in the call-by-name normal form of t. A pair of contexts $[\,] : \kappa \vdash_{\mathsf{ST}} C : \mathsf{o}$ and $[\,] : \kappa \vdash_{\mathsf{ST}} D : \kappa$ is called *linear* if $x : \kappa \vdash_{\mathsf{ST}} C[D^i[x]] : \mathsf{o}$ is linear for any $i \geq 0$ where x is a fresh variable that is not captured by the context applications.

▶ **Definition 3** (Higher-Order Grammar)**.** A *higher-order grammar* (or *grammar* for short) is a quadruple $(\Sigma, \mathcal{N}, \mathcal{R}, S)$, where (i) Σ is a ranked alphabet; (ii) $\mathcal{N}$ is a map from a finite set of non-terminals to their types; (iii) $\mathcal{R}$ is a finite set of *rewriting rules* of the form $A \to \lambda x_1. \cdots \lambda x_\ell.t$, where $\mathcal{N}(A) = \kappa_1 \to \cdots \to \kappa_\ell \to \mathsf{o}$, t is an applicative term, and $\mathcal{N}, x_1 : \kappa_1, \ldots, x_\ell : \kappa_\ell \vdash_{\mathsf{ST}} t : \mathsf{o}$ holds; (iv) S is a non-terminal called the *start symbol*, and $\mathcal{N}(S) = \mathsf{o}$. The *order* of a grammar $\mathcal{G}$ is the largest order of the types of non-terminals. We sometimes write $\Sigma_{\mathcal{G}}, \mathcal{N}_{\mathcal{G}}, \mathcal{R}_{\mathcal{G}}, S_{\mathcal{G}}$ for the four components of $\mathcal{G}$. We often write $A\, x_1 \cdots x_k \to t$ for the rule $A \to \lambda x_1. \cdots \lambda x_k.t$.

For a grammar $\mathcal{G} = (\Sigma, \mathcal{N}, \mathcal{R}, S)$, the rewriting relation $\longrightarrow_{\mathcal{G}}$ is defined by:

$$\frac{(A \to \lambda x_1. \cdots \lambda x_k.t) \in \mathcal{R}}{A\, t_1 \cdots t_k \longrightarrow_{\mathcal{G}} [t_1/x_1, \ldots, t_k/x_k]t} \qquad \frac{t_i \longrightarrow_{\mathcal{G}} t'_i \qquad i \in \{1, \ldots, k\} \qquad \Sigma(a) = k}{a\, t_1 \cdots t_k \longrightarrow_{\mathcal{G}} a\, t_1 \cdots t_{i-1}\, t'_i\, t_{i+1} \cdots t_k}$$

We write $\longrightarrow_{\mathcal{G}}^*$ for the reflexive transitive closure of $\longrightarrow_{\mathcal{G}}$. The *tree language generated by $\mathcal{G}$*, written $\mathcal{L}(\mathcal{G})$, is the set $\{\pi \mid S \longrightarrow_{\mathcal{G}}^* \pi\}$.

▶ Remark. An order-n grammar can also be represented as a ground closed order-n $\lambda^{\to,+}$-term extended with the Y-combinator such that $Y_\kappa x.t \longrightarrow [Y_\kappa x.t/x]t$. Conversely, any ground closed order-n $\lambda^{\to,+}$-term (extended with Y) can be represented as an equivalent order-n grammar.

The grammars defined above may also be viewed as generators of word languages.

▶ **Definition 4** (Word Alphabet / $\mathtt{br}$-Alphabet)**.** We call a ranked alphabet Σ a *word alphabet* if it has a special nullary terminal $\mathtt{e}$ and all the other terminals have arity 1; also we call a grammar $\mathcal{G}$ a *word grammar* if its alphabet is a word alphabet. For a tree $\pi = a_1(\cdots(a_n\,\mathtt{e})\cdots)$ of a word grammar, we define $\mathbf{word}(\pi) = a_1 \cdots a_n$. The *word language* generated by a word grammar $\mathcal{G}$, written $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$, is $\{\mathbf{word}(\pi) \mid \pi \in \mathcal{L}(\mathcal{G})\}$.

The frontier word of a tree π, written $\mathbf{leaves}(\pi)$, is the sequence of symbols in the leaves of π. It is defined inductively by: $\mathbf{leaves}(a) = a$ when $\Sigma(a) = 0$, and $\mathbf{leaves}(a\, \pi_1 \cdots \pi_k) = \mathbf{leaves}(\pi_1) \cdots \mathbf{leaves}(\pi_k)$ when $\Sigma(a) = k > 0$. The *frontier language* generated by $\mathcal{G}$, written

$\mathcal{L}_{\texttt{leaf}}(\mathcal{G})$, is the set: $\{\textbf{leaves}(\pi) \mid S \longrightarrow^*_{\mathcal{G}} \pi\}$. A `br`-*alphabet* is a ranked alphabet such that it has a special binary constant `br` and a special nullary constant `e` and the other constants are nullary. We consider `e` as the empty word ε: for a grammar with a `br`-alphabet, we also define $\mathcal{L}^{\varepsilon}_{\texttt{leaf}}(\mathcal{G}) := (\mathcal{L}_{\texttt{leaf}}(\mathcal{G}) \setminus \{\texttt{e}\}) \cup \{\varepsilon \mid \texttt{e} \in \mathcal{L}_{\texttt{leaf}}(\mathcal{G})\}$. We call a tree π an `e`-*free* `br`-*tree* if it is a tree of some `br`-alphabet but does not contain `e`.

We note that the classes of order-0, order-1, and order-2 word languages coincide with those of regular, context-free, and indexed languages, respectively [26].

2.2 Homeomorphic Embedding and Kruskal's Tree Theorem

In our main theorem, we use the notion of homeomorphic embedding for trees.

▶ **Definition 5** (Homeomorphic Embedding). Let Σ be an arbitrary ranked alphabet. The *homeomorphic embedding* order $\preceq$ between Σ-ranked trees[5] is inductively defined by the following rules:

$$\frac{\pi_i \preceq \pi'_i \quad (\text{for all } i \le k)}{a\,\pi_1 \cdots \pi_k \preceq a\,\pi'_1 \cdots \pi'_k}(k = \Sigma(a)) \qquad \frac{\pi \preceq \pi_i}{\pi \preceq a\,\pi_1 \cdots \pi_k}(k = \Sigma(a) > 0,\ i \in \{1, \dots, k\})$$

For example, $\texttt{br}\,\texttt{a}\,\texttt{b} \preceq \texttt{br}\,(\texttt{br}\,\texttt{a}\,\texttt{c})\,\texttt{b}$. We extend $\preceq$ to words: for $w = a_1 \cdots a_n$ and $w' = a'_1 \cdots a'_{n'}$, we define $w \preceq w'$ if $a_1(\cdots(a_n(\texttt{e}))) \preceq a'_1(\cdots(a'_{n'}(\texttt{e})))$, where a_i and a'_i are regarded as unary constants and `e` is a nullary constant (this order on words is nothing but the (scattered) subsequence relation). We write $\pi \prec \pi'$ if $\pi \preceq \pi'$ and $\pi' \not\preceq \pi$.

Next we explain a basic property on $\preceq$, Kruskal's tree theorem. A *quasi-order* (also called a *pre-order*) is a reflexive and transitive relation. A *well quasi-order* on a set S is a quasi-order $\le$ on S such that for any infinite sequence $(s_i)_i$ of elements in S there exist $j < k$ such that $s_j \le s_k$.

▶ **Proposition 6** (Kruskal's Tree Theorem [17]). *For any (finite) ranked alphabet Σ, the homeomorphic embedding $\preceq$ on Σ-ranked trees is a well quasi-order.*

2.3 Conjecture and Pumping Lemma for Higher-order Grammars

As explained in Section 1, our pumping lemma makes use of a conjecture on "higher-order" Kruskal's tree theorem, which is stated below.

▶ **Conjecture 7.** *There exists a family $(\preceq_\kappa)_\kappa$ of relations indexed by simple types such that*

- *$\preceq_\kappa$ is a well quasi-order on the set of closed $\lambda^{\to}$-terms of type κ modulo $\beta\eta$-equivalence; i.e., for an infinite sequence $t_1, t_2, \dots$ of closed $\lambda^{\to}$-terms of type κ, there exist $i < j$ such that $t_i \preceq_\kappa t_j$.*
- *$\preceq_{\mathsf{o}}$ is a conservative extension of $\preceq$, i.e., $t \preceq_{\mathsf{o}} t'$ if and only if $\mathcal{T}(t) \preceq \mathcal{T}(t')$.*
- *$(\preceq_\kappa)_\kappa$ is closed under applications, i.e., if $t \preceq_{\kappa_1 \to \kappa_2} t'$ and $s \preceq_{\kappa_1} s'$ then $t\,s \preceq_{\kappa_2} t'\,s'$.*

A candidate of $(\preceq_\kappa)_\kappa$ would be the logical relation induced from $\preceq$. Indeed, if we choose the logical relation as $(\preceq_\kappa)_\kappa$, the above conjecture holds up to order-2 (see Theorem 18 in Section 6).

Actually, for our pumping lemma, the following, slightly weaker property called the *periodicity* is sufficient.

[5] In the usual definition, a quasi order on labels (tree constructors) is assumed. Here we fix the quasi-order on labels to the identity relation.

▶ **Conjecture 8** (Periodicity). *There exists a family $(\preceq_\kappa)_\kappa$ indexed by simple types such that*
- $\preceq_\kappa$ *is a quasi-order on the set of closed $\lambda^{\rightarrow}$-terms of type κ modulo $\beta\eta$-equivalence.*
- *for any $\vdash_{\mathtt{ST}} t : \kappa \to \kappa$ and $\vdash_{\mathtt{ST}} s : \kappa$, there exist $i, j > 0$ such that*

$$t^i\, s \preceq_\kappa t^{i+j}\, s \preceq_\kappa t^{i+2j}\, s \preceq_\kappa \cdots .$$

- $\preceq_{\mathtt{o}}$ *is a conservative extension of $\preceq$.*
- $(\preceq_\kappa)_\kappa$ *is closed under applications.*

Note that Conjecture 7 implies Conjecture 8, since if the former holds, for the infinite sequence $(t^i\, s)_i$, there exist $i < i + j$ such that $t^i\, s \preceq_\kappa t^{i+j}\, s$, and then by the monotonicity of $u \mapsto t^j\, u$, we have $t^{i+kj}\, s \preceq_\kappa t^{i+(k+1)j}\, s$ for any $k \geq 0$.

We can now state our pumping lemma.

▶ **Theorem 9** (Pumping Lemma). *Assume that Conjecture 8 holds. Then, for any order-n tree grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G})$ is infinite, there exist an infinite sequence of trees $\pi_0, \pi_1, \pi_2, \ldots \in \mathcal{L}(\mathcal{G})$, and constants c, d such that:* (i) $\pi_0 \prec \pi_1 \prec \pi_2 \prec \cdots$, *and* (ii) $|\pi_i| \leq \mathbf{exp}_n(ci + d)$ *for each $i \geq 0$. Furthermore, we can drop the assumption on Conjecture 8 when $\mathcal{G}$ is of order up to 2.*

By the correspondence between order-n tree grammars and order-$(n+1)$ grammars [4, 1], we also have:

▶ **Corollary 10** (Pumping Lemma for Word Languages). *Assume that Conjecture 8 holds. Then, for any order-n word grammar $\mathcal{G}$ (where $n \geq 1$) such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G})$ is infinite, there exist an infinite sequence of words $w_0, w_1, w_2, \ldots \in \mathcal{L}_{\mathtt{w}}(\mathcal{G})$, and constants c, d such that:* (i) $w_0 \prec w_1 \prec w_2 \prec \cdots$, *and* (ii) $|w_i| \leq \mathbf{exp}_{n-1}(ci + d)$ *for each $i \geq 0$. Furthermore, we can drop the assumption on Conjecture 8 when $\mathcal{G}$ is of order up to 3.*

We sketch the overall structure of the proof of Theorem 9 below. Let $\mathcal{G}$ be an order-n tree grammar. By using the recent type system of Parys [23], if $\mathcal{L}(\mathcal{G})$ is infinite, we can construct order-n linear $\lambda^{\rightarrow}$-contexts C, D and an order-n $\lambda^{\rightarrow}$-term t such that $\{\mathcal{T}(C[D^i[t]]) \mid i \geq 0\}$ $(\subseteq \mathcal{L}(\mathcal{G}))$ is infinite. It then suffices to show that there exist constants p and q such that $\mathcal{T}(C[D^p[t]]) \prec \mathcal{T}(C[D^{p+q}[t]]) \prec \mathcal{T}(C[D^{p+2q}[t]]) \prec \cdots$. The bound $\mathcal{T}(C[D^{p+iq}]) \leq \mathbf{exp}_n(c + id)$ would then follow immediately from the standard result on an upper-bound on the size of β-normal forms. Actually, assuming Conjecture 8, we can easily deduce $\mathcal{T}(C[D^p[t]]) \preceq \mathcal{T}(C[D^{p+q}[t]]) \preceq \mathcal{T}(C[D^{p+2q}[t]]) \preceq \cdots$. Thus, the main remaining difficulty is to show that the "strict" inequality holds periodically. To this end, we prove it by induction on the order, by making use of three ingredients: an extension of the result of Parys' type system (again) [23], an extension of our previous work on a translation from word languages to tree languages [1], and Conjecture 8. In Sections 3 and 4, we derive corollaries from the results of Parys' and our previous work respectively. We then provide the proof of Theorem 9 (except the statement "Furthermore, ...") in Section 5. We then, in Section 6, discharge the assumption on Conjecture 8 for order up to 2, by proving Conjecture 7 for order up to 2.

3 Corollaries of Parys' Results

Parys [23] developed an intersection type system with judgments of the form $\Gamma \vdash s : \tau \triangleright c$, where s is a term of a simply-typed, infinitary λ-calculus (that corresponds to the λY-calculus) extended with choice, and c is a natural number. He proved that for any order-n closed ground term s, (i) $\emptyset \vdash s : \tau \triangleright c$ implies that s can be reduced to a tree π such that $c \leq |\pi|$, and (ii) if s can be reduced to a tree π, then $\emptyset \vdash s : \tau \triangleright c$ holds for some c such that $|\pi| \leq \mathbf{exp}_n(c)$.

Let $\mathcal{G}$ be an order-n tree grammar and S be its start symbol. By Parys' result,[6] if $\mathcal{L}(\mathcal{G})$ is infinite, there exists a derivation for $\emptyset \vdash S : \mathtt{o} \triangleright c_1 + c_2 + c_3$ in which $\Theta \vdash A : \gamma \triangleright c_1 + c_2$ is derived from $\Theta \vdash A : \gamma \triangleright c_1$ for some non-terminal A. Thus, by "pumping" the derivation of $\Theta \vdash A : \gamma \triangleright c_1 + c_2$ from $\Theta \vdash A : \gamma \triangleright c_1$, we obtain a derivation for $\emptyset \vdash S : \mathtt{o} \triangleright c_1 + kc_2 + c_3$ for any $k \geq 0$. From the derivation, we obtain a $\lambda^{\to}$-term t and $\lambda^{\to}$-contexts C, D of at most order-n, such that $C[D^k[t]]$ generates a tree π_k such that $c_1 + kc_2 + c_3 \leq |\pi_k|$. By further refining the argument above (see the full version for details), we can also ensure that the pair (C, D) is linear. Thus, we obtain the following lemma.

▶ **Lemma 11.** *Given an order-n tree grammar $\mathcal{G}$ such that $\mathcal{L}(\mathcal{G})$ is infinite, there exist order-n linear $\lambda^{\to}$-contexts C, D, and an order-n $\lambda^{\to}$-term t such that:*
1. $\{\mathcal{T}(C[D^k[t]]) \mid k \geq 1\} \subseteq \mathcal{L}(\mathcal{G})$,
2. $\{\mathcal{T}(C[D^{\ell_k}[t]]) \mid k \geq 1\}$ *is infinite for any strictly increasing sequence* $(\ell_k)_k$.

By slightly modifying Parys' type system, we can also reason about the length of a particular path of a tree. Let us annotate each constructor a as $a^{\langle i \rangle}$, where $0 \leq i \leq \Sigma(a)$. We call i a *direction*. We define $|\pi|_{\mathrm{p}}$ by:

$$|a^{\langle 0 \rangle}\,\pi_1\,\cdots\,\pi_k|_{\mathrm{p}} = 1 \qquad |a^{\langle i \rangle}\,\pi_1\,\cdots\,\pi_k|_{\mathrm{p}} = |\pi_i|_{\mathrm{p}} + 1 \quad (1 \leq i \leq k).$$

We define **rmdir** as the function that removes all the direction annotations.

▶ **Lemma 12.** *For any order-n linear $\lambda^{\to}$-contexts C, D and any order-n $\lambda^{\to}$-term t such that $\{\mathcal{T}(C[D^k[t]]) \mid k \geq 1\}$ is infinite, there exist direction-annotated order-n linear $\lambda^{\to}$-contexts G, H, a direction-annotated order-n $\lambda^{\to}$-term u, and $p, q > 0$ such that*
1. $\mathbf{rmdir}(\mathcal{T}(G[H^k[u]])) = \mathcal{T}(C[D^{pk+q}[t]])$ *for any* $k \geq 1$,
2. $\{|\mathcal{T}(G[H^{\ell_k}[u]])|_{\mathrm{p}} \mid k \geq 1\}$ *is infinite for any strictly increasing sequence* $(\ell_k)_k$.

4 Word to Frontier Transformation

We have an "order-decreasing" transformation [1] that transforms an order-$(n+1)$ word grammar $\mathcal{G}$ to an order-n tree grammar $\mathcal{G}'$ (with a $\mathtt{br}$-alphabet) such that $\mathcal{L}_{\mathtt{w}}(\mathcal{G}) = \mathcal{L}^{\varepsilon}_{\mathtt{leaf}}(\mathcal{G}')$. We use this as a method for induction on order; this method was originally suggested by Damm [4] for safe languages.

The transformation in the present paper has been modified from the original one in [1]. On the one hand, the current transformation is a specialized version in that we apply the transformation only to $\lambda^{\to}$-terms instead of terms of (non-deterministic) grammars. On the other hand, the current transformation has been strengthened in that the transformation preserves linearity. Due to the preservation of linearity, a *single*-hole context is transformed to a *single*-hole context, and the uniqueness of an occurrence of $[\,]$ will be utilized for the calculation of the size of "pumped trees" in Lemma 16.

The definition of the current transformation is given just by translating the transformation rules in [1] by following the idea of the embedding of $\lambda^{\to}$-terms into grammars. For the detailed definition, see the full version. By using this transformation, we have:

▶ **Lemma 13.** *Given order-n $\lambda^{\to}$-contexts C, D, and an order-n $\lambda^{\to}$-term t such that*
- *the constants in C, D, t are in a word alphabet,*

[6] See Section 6 of [23]. Parys considered a λ-calculus with infinite regular terms, but the result can be easily adapted to terms of grammars.

- $\{\mathcal{T}(C[D^{\ell_i}[t]]) \mid i \geq 0\}$ *is infinite for any strictly increasing sequence* $(\ell_i)_i$, *and*
- C *and* D *are linear,*

there exist order-$(n-1)$ $\lambda^{\rightarrow}$*-contexts* G, H, *order-*$(n-1)$ $\lambda^{\rightarrow}$*-term* u, *and some constant numbers* c, $d \geq 1$ *such that*

- *the constants in* G, H, u *are in a* `br`*-alphabet*
- *for* $i \geq 0$, $\mathcal{T}(G[H^i[u]])$ *is either an* `e`*-free* `br`*-tree or* `e`, *and*

$$\textbf{word}(\mathcal{T}(C[D^{ci+d}[t]])) = \begin{cases} \varepsilon & (\mathcal{T}(G[H^i[u]]) = \mathtt{e}) \\ \textbf{\textit{leaves}}(\mathcal{T}(G[H^i[u]])) & (\mathcal{T}(G[H^i[u]]) \neq \mathtt{e}) \end{cases}$$

- G *and* H *are linear.*

Proof. The preservation of meaning (the second condition) follows as a corollary of a theorem in [1]. Also, the preservation of linearity (the third condition) can be proved in a manner similar to the proof of the preservation of meaning in [1], using a kind of subject-reduction. See the full version for the detail. ◀

5 Proof of the Main Theorem

We first prepare some lemmas.

▶ **Lemma 14.** *For* `e`*-free* `br`*-trees* π *and* π', *if* $\pi \prec \pi'$ *then* $\textbf{\textit{leaves}}(\pi) \prec \textbf{\textit{leaves}}(\pi')$.

Proof. We can show that $\pi \preceq \pi'$ implies $\textbf{leaves}(\pi) \preceq \textbf{leaves}(\pi')$ and then the statement, both by straightforward induction on the derivation of $\pi \preceq \pi'$. ◀

▶ Remark. The above lemma does not necessarily hold for an arbitrary ranked alphabet, especially that with a unary constant; e.g., `a e` $\prec$ `a (a e)` but their leaves are both `e`. Also, it does not hold if a tree contains `e` and if we regard `e` as ε in the leaves word; e.g., for `br a b` $\prec$ `br (br a e) b`, their leaves are `ab` $\prec$ `aeb`, but if we regard `e` as ε then `ab` $\not\prec$ `ab`.

▶ **Lemma 15.** *For direction-annotated trees* π *and* π', *if* $\pi \prec \pi'$ *then* $\textbf{rmdir}(\pi) \prec \textbf{rmdir}(\pi')$.

Proof. We can show that $\pi \preceq \pi'$ implies $\textbf{rmdir}(\pi) \preceq \textbf{rmdir}(\pi')$ and then the statement, both by straightforward induction on the derivation of $\pi \preceq \pi'$. ◀

Now, we prove the following lemma (Lemma 16) by the induction on order. Theorem 9 (except the last statement) will then follow as an immediate corollary of Lemmas 11 and 16.

▶ **Lemma 16.** *Assume that the statement of Conjecture 8 is true. For any order-*n *linear* $\lambda^{\rightarrow}$*-contexts* C, D *and any order-*n $\lambda^{\rightarrow}$*-term* t *such that* $\{\mathcal{T}(C[D^i[t]]) \mid i \geq 1\}$ *is infinite, there exist* c, d, j, $k \geq 1$ *such that*

- $\mathcal{T}(C[D^j[t]]) \prec \mathcal{T}(C[D^{j+k}[t]]) \prec \mathcal{T}(C[D^{j+2k}[t]]) \prec \cdots$
- $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci+d) \qquad (i = 0, 1, \dots)$

Proof. The proof proceeds by induction on n. The case $n = 0$ is clear, and we discuss the case $n > 0$ below. By Lemma 12, from C, D, and t, we obtain direction-annotated order-n linear $\lambda^{\rightarrow}$-contexts G, H, a direction-annotated order-n $\lambda^{\rightarrow}$-term u, and j_0, $k_0 > 0$ such that

$$\textbf{rmdir}(\mathcal{T}(G[H^i[u]])) = \mathcal{T}(C[D^{j_0+ik_0}[t]]) \text{ for any } i \geq 1 \tag{1}$$

$$\{|\mathcal{T}(G[H^{\ell_i}[u]])|_{\mathrm{p}} \mid i \geq 1\} \text{ is infinite for any strictly increasing sequence } (\ell_i)_i. \tag{2}$$

Next we transform G, H, and u by choosing a path according to directions, i.e., we define G_{p}, H_{p}, and u_{p} as the contexts/term obtained from G, H, and u by replacing each $a^{\langle i\rangle}$ with: (i) $\lambda x_1 \dots x_\ell . a_i x_i$ if $i > 0$ or (ii) $\lambda x_1 \dots x_\ell . \mathtt{e}$ if $i = 0$, where $\ell = \Sigma(a)$ and a_i is a fresh unary constant. For any $i \geq 0$,

$$|\mathcal{T}(G[H^i[u]])|_{\mathrm{p}} = |\mathbf{word}(\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]))| + 1. \tag{3}$$

We also define a function **path** on trees annotated with directions, by the following induction: $\mathbf{path}(a^{\langle i\rangle}\, \pi_1 \cdots \pi_\ell) = a_i\, \mathbf{path}(\pi_i)$ if $i > 0$ and $\mathbf{path}(a^{\langle 0\rangle}\, \pi_1 \cdots \pi_\ell) = \mathtt{e}$. Then for any $i \geq 0$,

$$\mathbf{path}(\mathcal{T}(G[H^i[u]])) = \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]). \tag{4}$$

By (2) and (3), $\{\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{\ell_i}[u_{\mathrm{p}}]]) \mid i \geq 0\}$ is infinite for any strictly increasing sequence $(\ell_i)_i$. Also, the transformation from G, H to G_{p}, H_{p} preserves the linearity, because: let N be the normal form of $G[H^i[x]]$ where x is fresh, and N_{p} be the term obtained by applying this transformation to N; then $G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[x]] \longrightarrow^* N_{\mathrm{p}}$, and by the infiniteness of $\{\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^i[u_{\mathrm{p}}]]) \mid i \geq 0\}$, N_{p} must contain x, which implies N_{p} is a linear normal form.

Now we decrease the order by using the transformation in Section 4. By Lemma 13 to G_{p}, H_{p}, and u_{p}, there exist order-$(n-1)$ linear $\lambda^{\to}$-contexts G_l, H_l, an order-$(n-1)$ $\lambda^{\to}$-term u_l, and some constant numbers c', $d' \geq 1$ such that, for any $i \geq 0$, $\mathcal{T}(G_l[{H_l}^i[u_l]])$ is either an $\mathtt{e}$-free $\mathtt{br}$-tree or $\mathtt{e}$, and

$$\mathbf{word}(\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'i+d'}[u_{\mathrm{p}}]])) = \begin{cases} \varepsilon & (\mathcal{T}(G_l[{H_l}^i[u_l]]) = \mathtt{e}) \\ \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^i[u_l]])) & (\mathcal{T}(G_l[{H_l}^i[u_l]]) \neq \mathtt{e}). \end{cases} \tag{5}$$

By (2), (3), and (5), $\{\mathcal{T}(G_l[{H_l}^i[u_l]]) \mid i \geq 1\}$ is also infinite.

By the induction hypothesis, there exist j_1 and k_1 such that

$$\mathcal{T}(G_l[{H_l}^{j_1}[u_l]]) \prec \mathcal{T}(G_l[{H_l}^{j_1+k_1}[u_l]]) \prec \mathcal{T}(G_l[{H_l}^{j_1+2k_1}[u_l]]) \prec \cdots .$$

Hence by Lemma 14, we have

$$\mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1}[u_l]])) \prec \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1+k_1}[u_l]])) \prec \mathbf{leaves}(\mathcal{T}(G_l[{H_l}^{j_1+2k_1}[u_l]])) \prec \cdots .$$

Then by (5), we have

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'j_1+d'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'(j_1+k_1)+d'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{c'(j_1+2k_1)+d'}[u_{\mathrm{p}}]]) \prec \cdots .$$

Let $j_1' = c'j_1 + d'$ and $k_1' = c'k_1$; then

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'+k_1'}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_1'+2k_1'}[u_{\mathrm{p}}]]) \prec \cdots . \tag{6}$$

Now, by Conjecture 8, there exist $j_2 \geq 0$ and $k_2 > 0$ such that

$$H^{j_2}[u] \preceq_\kappa H^{j_2+k_2}[u] \preceq_\kappa H^{j_2+2k_2}[u] \preceq_\kappa \cdots . \tag{7}$$

Let j_3 be the least j_3 such that $j_3 = j_1' + i_3\, k_1' = j_2 + m_0$ for some i_3 and m_0, and k_3 be the least common multiple of k_1' and k_2, whence $k_3 = m_1 k_1' = m_2 k_2$ for some m_1 and m_2. Then since the mapping $s \mapsto \mathcal{T}(G[H^{m_0}[s]])$ is monotonic, from (7) we have:

$$\mathcal{T}(G[H^{j_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+k_2}[u]]) \preceq \mathcal{T}(G[H^{j_3+2k_2}[u]]) \preceq \cdots .$$

Since $j_3 + ik_3 = j_3 + (im_2)k_2$, we have

$$\mathcal{T}(G[H^{j_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+k_3}[u]]) \preceq \mathcal{T}(G[H^{j_3+2k_3}[u]]) \preceq \cdots . \tag{8}$$

Also, since $j_3 + ik_3 = j'_1 + (i_3 + im_1)k'_1$, from (6) we have

$$\mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3+k_3}[u_{\mathrm{p}}]]) \prec \mathcal{T}(G_{\mathrm{p}}[{H_{\mathrm{p}}}^{j_3+2k_3}[u_{\mathrm{p}}]]) \prec \cdots . \tag{9}$$

Thus, from (4), (8), and (9) we obtain

$$\mathcal{T}(G[H^{j_3}[u]]) \prec \mathcal{T}(G[H^{j_3+k_3}[u]]) \prec \mathcal{T}(G[H^{j_3+2k_3}[u]]) \prec \cdots . \tag{10}$$

By applying **rmdir** to this sequence, and by (1) and Lemma 15, we have

$$\mathcal{T}(C[D^{j_0+j_3k_0}[t]]) \prec \mathcal{T}(C[D^{j_0+(j_3+k_3)k_0}[t]]) \prec \mathcal{T}(C[D^{j_0+(j_3+2k_3)k_0}[t]]) \prec \cdots . \tag{11}$$

We define $j = j_0 + k_0 j_3$ and $k = k_0 k_3$; then we obtain

$$\mathcal{T}(C[D^{j}[t]]) \prec \mathcal{T}(C[D^{j+k}[t]]) \prec \mathcal{T}(C[D^{j+2k}[t]]) \prec \cdots .$$

Finally, we show that $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci + d)$ for some c and d. Since C and D are single-hole contexts, $|C[D^{j+ik}[t]]| = |C| + (j + ik)|D| + |t|$. Let $c = k|D|$ and $d = |C| + j|D| + |t|$; then $|C[D^{j+ik}[t]]| = ci + d$. It is well-known that, for an order-n $\lambda^{\to}$-term s, we have $|\mathcal{T}(s)| \leq \mathbf{exp}_n(|s|)$ (see, e.g., [25, Lemma 3]). Thus, we have $|\mathcal{T}(C[D^{j+ik}[t]])| \leq \mathbf{exp}_n(ci + d)$. ◀

The step obtaining (10) (the steps using Lemma 14 and obtaining (11), resp.) indicates why we need to require $\mathcal{T}(C[D^{j+ik}[t]]) \prec \mathcal{T}(C[D^{j+i'k}[t]])$ for any $i < i'$ rather than $|\mathcal{T}(C[D^{j+ik}[t]])| < |\mathcal{T}(C[D^{j+i'k}[t]])|$ ($\mathcal{T}(C[D^{j+ik}[t]]) \neq \mathcal{T}(C[D^{j+i'k}[t]])$, resp.) to make the induction work.

6 Second-order Kruskal's theorem

In this section, we prove Conjecture 7 (hence also Conjecture 8) up to order-2. First, we extend the homeomorphic embedding $\preceq$ on trees to a family of relations $\preceq_\kappa$ by using logical relation: (i) $t_1 \preceq_{\mathsf{o}} t_2$ if $\emptyset \vdash_{\mathsf{ST}} t_1 : \mathsf{o}$, $\emptyset \vdash_{\mathsf{ST}} t_2 : \mathsf{o}$, and $\mathcal{T}(t_1) \preceq \mathcal{T}(t_2)$. (ii) $t_1 \preceq_{\kappa_1 \to \kappa_2} t_2$ if $\emptyset \vdash_{\mathsf{ST}} t_1 : \kappa_1 \to \kappa_2$, $\emptyset \vdash_{\mathsf{ST}} t_2 : \kappa_1 \to \kappa_2$, and $t_1 s_1 \preceq_{\kappa_2} t_2 s_2$ holds for every s_1, s_2 such that $s_1 \preceq_{\kappa_1} s_2$. We often omit the subscript κ and just write $\preceq$ for $\preceq_\kappa$. We also write $x_1 : \kappa_1, \ldots, x_k : \kappa_k \models t \preceq_\kappa t'$ if $[s_1/x_1, \ldots, s_k/x_k]t \preceq_\kappa [s'_1/x_1, \ldots, s'_k/x_k]t'$ for every $s_1, \ldots, s_k, s'_1, \ldots, s'_k$ such that $s_i \preceq_{\kappa_i} s'_i$.

The relation $\preceq_\kappa$ is well-defined for $\beta\eta$-equivalence classes, and by the abstraction lemma of logical relation, it turns out that the relation $\preceq_\kappa$ is a pre-order for any κ (see the full version for these). Note that the relation is also preserved by applications by the definition of the logical relation. It remains to show that $\preceq_\kappa$ is a well quasi-order for κ of order up to 2.

For ℓ-ary terminal a and $k \geq \ell$, we write $\mathbf{CTerms}_{a,k}$ for the set of terms

$$\{\lambda x_1. \cdots \lambda x_k. a\, x_{i_1} \, \ldots \, x_{i_\ell} \mid i_1 \cdots i_\ell \text{ is a subsequence of } 1 \cdots k\}.$$

We define $\mathsf{o}^0 \to \mathsf{o} \ := \ \mathsf{o}$ and $\mathsf{o}^{n+1} \to \mathsf{o} \ := \ \mathsf{o} \to (\mathsf{o}^n \to \mathsf{o})$.

The following lemma allows us to reduce $t \preceq_\kappa t'$ on any order-2 type κ to (finitely many instances of) that on order-0 type o.

▶ **Lemma 17.** *Let Σ be a ranked alphabet; κ be $(\mathsf{o}^{k_1} \to \mathsf{o}) \to \cdots \to (\mathsf{o}^{k_m} \to \mathsf{o}) \to \mathsf{o}$; a_i^j be a j-ary terminal not in Σ for $1 \leq i \leq m$ and $0 \leq j \leq k_i$; and t, t' be $\lambda^{\to}$-terms whose type is κ and whose terminals are in Σ. Then $t \preceq_\kappa t'$ if and only if $t\, u_1 \, \ldots \, u_m \preceq_{\mathsf{o}} t'\, u_1 \, \ldots \, u_m$ for every $u_i \in \cup_{j \leq k_i} \mathbf{CTerms}_{a_i^j, k_i}$.*

Proof. The "only if" direction is trivial by the definition of $\preceq_\kappa$. To show the opposite, assume the latter holds. We need to show that $t\, s_1\, \dots\, s_m \preceq_{\mathtt{o}} t'\, s_1\, \dots\, s_m$ holds for every combination of $s_1, \dots, s_m$ such that $\vdash_{\mathtt{ST}} s_i : \kappa_i$ for each i. Without loss of generality, we can assume that t, t', $s_1, \dots, s_m$ are $\beta\eta$ long normal forms, and hence that

$$\begin{array}{lr} t = \lambda f_1.\cdots\lambda f_m.t_0 & f_1 : \mathtt{o}^{k_1} \to \mathtt{o}, \dots, f_m : \mathtt{o}^{k_m} \to \mathtt{o} \vdash_{\mathtt{ST}} t_0 : \mathtt{o} \\ t' = \lambda f_1.\cdots\lambda f_m.t'_0 & f_1 : \mathtt{o}^{k_1} \to \mathtt{o}, \dots, f_m : \mathtt{o}^{k_m} \to \mathtt{o} \vdash_{\mathtt{ST}} t'_0 : \mathtt{o} \\ s_i = \lambda x_1.\dots\lambda x_{k_i}.s_{i,0} & x_1 : \mathtt{o}, \dots, x_{k_i} : \mathtt{o} \vdash_{\mathtt{ST}} s_{i,0} : \mathtt{o} \quad \text{(for each } i) \end{array}$$

For each $i \leq m$, let $\mathbf{FV}(s_{i,0}) = \{x_{q(i,1)}, \dots, x_{q(i,\ell_i)}\}$, and $u_i \in \mathbf{CTerms}_{a_i^{\ell_i},k_i}$ be the term $\lambda x_1.\cdots\lambda x_{k_i}.a_i^{\ell_i}\, x_{q(i,1)}\, \cdots\, x_{q(i,\ell_i)}$. Let θ and θ' be the substitutions $[u_1/f_1, \dots, u_m/f_m]$ and $[s_1/f_1, \dots, s_m/f_m]$ respectively. It suffices to show that $\theta t_0 \preceq_{\mathtt{o}} \theta t'_0$ implies $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_0$, which we prove by induction on $|t'_0|$.

By the condition $f_1 : \mathtt{o}^{k_1} \to \mathtt{o}, \dots, f_m : \mathtt{o}^{k_m} \to \mathtt{o} \vdash_{\mathtt{ST}} t_0 : \mathtt{o}$, t_0 must be of the form $h\, t_1\, \cdots\, t_\ell$ where h is f_i or a terminal a in Σ, and ℓ may be 0. Then we have

$$\mathcal{T}(\theta t_0) = \begin{cases} a\mathcal{T}(\theta t_1)\cdots\mathcal{T}(\theta t_\ell) & (h = a) \\ a_i^{\ell_i}\mathcal{T}(\theta t_{q(i,1)})\cdots\mathcal{T}(\theta t_{q(i,\ell_i)}) & (h = f_i) \end{cases}$$

Similarly, t'_0 must be of the form $h'\, t'_1\, \cdots\, t'_{\ell'}$ and the corresponding equality on $\mathcal{T}(\theta t'_0)$ holds. By the assumption $\theta t_0 \preceq_{\mathtt{o}} \theta t'_0$, we have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_0)$. We perform case analysis on the rule used for deriving $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_0)$ (recall Definition 5).

- Case of the first rule: In this case, the roots of $\mathcal{T}(\theta t_0)$ and $\mathcal{T}(\theta t'_0)$ are the same and hence $h = h'$ and $\ell = \ell'$. We further perform case analysis on h.
  - Case $h = a$: For $1 \leq j \leq \ell$, since $\mathcal{T}(\theta t_j) \preceq \mathcal{T}(\theta t'_j)$, by induction hypothesis, we have $\theta' t_j \preceq_{\mathtt{o}} \theta' t'_j$. Hence $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_0$.
  - Case $h = f_i$: For $1 \leq j \leq \ell_i$, since $\mathcal{T}(\theta t_{q(i,j)}) \preceq \mathcal{T}(\theta t'_{q(i,j)})$, by induction hypothesis, we have $\theta' t_{q(i,j)} \preceq_{\mathtt{o}} \theta' t'_{q(i,j)}$. Hence, $[\theta' t_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0} \preceq_{\mathtt{o}} [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$. By the definition of $q(i,j)$, $\theta' t_0 \longrightarrow [\theta' t_j/x_j]_{j \leq k_i} s_{i,0} = [\theta' t_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$, and similarly, $\theta' t'_0 \longrightarrow [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$; hence we have $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_0$.
- Case of the second rule: We further perform case analysis on h'.
  - Case $h' = a$: We have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_p)$ for some $1 \leq p \leq \ell'$. Hence by induction hypothesis, we have $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_p$, and then $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_0$.
  - Case $h' = f_i$: We have $\mathcal{T}(\theta t_0) \preceq \mathcal{T}(\theta t'_{q(i,p)})$ for some $1 \leq p \leq \ell_i$. Hence by induction hypothesis, we have $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_{q(i,p)}$. Also, by the definition of $q(i,p)$, $x_{q(i,p)}$ occurs in $s_{i,0}$. Since $s_{i,0}$ is a $\beta\eta$ long normal form of order-0, the order-0 variable $x_{q(i,p)}$ occurs as a leaf of $s_{i,0}$; hence $\mathcal{T}(\theta' t'_{q(i,p)}) \preceq [\mathcal{T}(\theta' t'_{q(i,j)})/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$. Therefore $\theta' t_0 \preceq_{\mathtt{o}} [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$. Since $\theta' t'_0 \longrightarrow [\theta' t'_{q(i,j)}/x_{q(i,j)}]_{j \leq \ell_i} s_{i,0}$, we have $\theta' t_0 \preceq_{\mathtt{o}} \theta' t'_0$. ◀

As a corollary, we obtain a second-order version of Kruskal's tree theorem.

▶ **Theorem 18.** *Let Σ be a ranked alphabet, κ be an at most order-2 type, and $t_0, t_1, t_2, \dots$ be an infinite sequence of $\lambda^{\to}$-terms whose type is κ and whose terminals are in Σ. Then, there exist $i < j$ such that $t_i \preceq_\kappa t_j$.*

Proof. Since κ is at most order-2, it must be of the form $(\mathtt{o}^{k_1} \to \mathtt{o}) \to \cdots \to (\mathtt{o}^{k_m} \to \mathtt{o}) \to \mathtt{o}$. Let a_i^j be a j-ary terminal not in Σ for $1 \leq i \leq m$ and $0 \leq j \leq k_i$; $(\cup_{j \leq k_1} \mathbf{CTerms}_{a_1^j,k_1}) \times \cdots \times (\cup_{j \leq k_m} \mathbf{CTerms}_{a_m^j,k_m})$ be $\{(u_{1,1}, \dots, u_{1,m}), \dots, (u_{p,1}, \dots, u_{p,m})\}$; b be a p-ary terminal not in $\Sigma \cup \{a_i^j \mid 1 \leq i \leq m,\ 0 \leq j \leq k_i\}$; and s_i be the term $b\,(t_i\, u_{1,1}\, \cdots\, u_{1,m})\, \cdots\, (t_i\, u_{p,1}\, \cdots\, u_{p,m})$

for each $i \in \{0, 1, 2, \ldots\}$. Since the set of terminals in $s_0, s_1, s_2, \ldots$ is finite, by Kruskal's tree theorem, there exist i, j such that $s_i \preceq_{\mathsf{o}} s_j$ and $i < j$. Since b occurs just at the root of s_k for each k, $s_i \preceq_{\mathsf{o}} s_j$ implies $t_i\, u_{k,1} \cdots u_{k,m} \preceq_{\mathsf{o}} t_j\, u_{k,1} \cdots u_{k,m}$ for every $k \in \{1, \ldots, p\}$. Thus, by Lemma 17, we have $t_i \preceq_{\kappa} t_j$ as required. ◀

7 Related Work

As mentioned in Section 1, to our knowledge, pumping lemmas for higher-order word languages have been established only up to order-2 [7], whereas we have proved (unconditionally) a pumping lemma for order-2 tree languages and order-3 word languages. Hayashi's pumping lemma for indexed languages (i.e., order-2 word languages) is already quite complex, and it is unclear how to generalize it to arbitrary orders. In contrast, our proof of a pumping lemma works for arbitrary orders, although it relies on the conjecture on higher-order Kruskal's tree theorem. Parys [21] and Kobayashi [12] studied pumping lemmas for collapsible pushdown automata and higher-order recursion schemes respectively. Unfortunately, they are not applicable to word/tree *languages* generated by (non-deterministic) grammars.

As also mentioned in Section 1, the strictness of hierarchy of higher-order word languages has already been shown by using a complexity argument [5, 8]. We can use our pumping lemma (if the conjecture is discharged) to obtain a simple alternative proof of the strictness, using the language $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$ as a witness of the separation between the classes of order-$(n+1)$ word languages and order-n word languages. In fact, the pumping lemma would imply that there is no order-n grammar that generates $\{a^{\mathbf{exp}_n(k)} \mid k \geq 0\}$, whereas an order-$(n+1)$ grammar that generates the same language can be easily constructed.

We are not aware of studies of the higher-order version of Kruskal's tree theorem (Conjecture 7) or the periodicity of tree functions expressed by the simply-typed λ-calculus (Conjecture 8), which seem to be of independent interest. Zaionc [27, 28] characterized the class of (first-order) word/tree functions definable in the simply-typed λ-calculus. To obtain higher-order Kruskal's tree theorem, we may need some characterization of *higher-order* definable tree functions instead.

We have heavily used the results of Parys' work [23] and our own previous work [1], which both use intersection types for studying properties of higher-order languages. Other uses of intersection types in studying higher-order grammars/languages are found in [10, 15, 22, 12, 3, 14, 13].

8 Conclusion

We have proved a pumping lemma for higher-order languages of arbitrary orders, modulo the assumption that a higher-order version of Kruskal's tree theorem holds. We have also proved the assumption indeed holds for the second-order case, yielding a pumping lemma for order-2 tree languages and order-3 word languages. Proving (or disproving) the higher-order Kruskal's tree theorem is left for future work.

Acknowledgments. We would like to thank Pawel Parys for discussions on his type system, and anonymous referees for useful comments.

References

1 Kazuyuki Asada and Naoki Kobayashi. On Word and Frontier Languages of Unsafe Higher-Order Grammars. In *43rd International Colloquium on Automata, Languages, and Pro-*

gramming (ICALP 2016), volume 55 of *LIPIcs*, pages 111:1–111:13. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2016.
2 Yehoshua Bar-Hillel, Micha A. Perles, and Eli Shamir. On formal properties of simple phrase structure grammars. *Z. Phonetik Sprachwiss. und Kommunikat.*, 14:143–172, 1961.
3 Lorenzo Clemente, Pawel Parys, Sylvain Salvati, and Igor Walukiewicz. The diagonal problem for higher-order recusion schemes is decidable. In *Proceedings of LICS 2016*, 2016.
4 Werner Damm. The IO- and OI-hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.
5 Joost Engelfriet. Iterated stack automata and complexity classes. *Info. Comput.*, 95(1):21–75, 1991.
6 Joost Engelfriet and Heiko Vogler. High level tree transducers and iterated pushdown tree transducers. *Acta Inf.*, 26(1/2):131–192, 1988.
7 Takeshi Hayashi. On derivation trees of indexed grammars – an extension of the uvwxy-theorem. *Publ. RIMS, Kyoto Univ.*, pages 61–92, 1973.
8 Alexander Kartzow. Personal communication, via Pawel Parys, 2013.
9 Teodor Knapik, Damian Niwinski, and Pawel Urzyczyn. Deciding monadic theories of hyperalgebraic trees. In *TLCA 2001*, volume 2044 of *LNCS*, pages 253–267. Springer, 2001.
10 Naoki Kobayashi. Types and higher-order recursion schemes for verification of higher-order programs. In *Proc. of POPL*, pages 416–428. ACM Press, 2009.
11 Naoki Kobayashi. Model checking higher-order programs. *Journal of the ACM*, 60(3), 2013.
12 Naoki Kobayashi. Pumping by typing. In *Proceedings of LICS 2013*, pages 398–407. IEEE Computer Society, 2013.
13 Naoki Kobayashi, Kazuhiro Inaba, and Takeshi Tsukada. Unsafe order-2 tree languages are context-sensitive. In *Proceedings of FoSSaCS 2014*, volume 8412 of *LNCS*, pages 149–163. Springer, 2014.
14 Naoki Kobayashi, Kazutaka Matsuda, Ayumi Shinohara, and Kazuya Yaguchi. Functional programs as compressed data. *Higher-Order and Symbolic Computation*, 2013.
15 Naoki Kobayashi and C.-H. Luke Ong. A type system equivalent to the modal mu-calculus model checking of higher-order recursion schemes. In *Proceedings of LICS 2009*, pages 179–188. IEEE Computer Society Press, 2009.
16 Gregory M. Kobele and Sylvain Salvati. The IO and OI hierarchies revisited. *Inf. Comput.*, 243:205–221, 2015.
17 J. B. Kruskal. Well-quasi-ordering, the tree theorem, and vazsonyi's conjecture. *Transactions of the American Mathematical Society*, 95(2):210–225, 1960. URL: http://www.jstor.org/stable/1993287.
18 A. N. Maslov. The hierarchy of indexed languages of an arbitrary level. *Soviet Math. Dokl.*, 15:1170–1174, 1974.
19 C. St. J. A. Nash-Williams. On well-quasi-ordering finite trees. *Mathematical Proceedings of the Cambridge Philosophical Society*, 59(4):833–835, 1963.
20 C.-H. Luke Ong. On model-checking trees generated by higher-order recursion schemes. In *LICS 2006*, pages 81–90. IEEE Computer Society Press, 2006.
21 Pawel Parys. A pumping lemma for pushdown graphs of any level. In *Proceedings of STACS 2012*, volume 14 of *LIPIcs*, pages 54–65. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.
22 Pawel Parys. How many numbers can a lambda-term contain? In *Proceedings of FLOPS 2014*, volume 8475 of *LNCS*, pages 302–318. Springer, 2014. doi:10.1007/978-3-319-07151-0_19.
23 Pawel Parys. Intersection types and counting. *CoRR*, abs/1701.05303, 2017. A shorter version will appear in Post-proceedings of ITRS 2016. URL: http://arxiv.org/abs/1701.05303.

24 Sylvain Salvati and Igor Walukiewicz. Krivine machines and higher-order schemes. In *Proceedings of ICALP 2011*, volume 6756 of *LNCS*, pages 162–173. Springer, 2011.

25 Kazushige Terui. Semantic evaluation, intersection types and complexity of simply typed lambda calculus. In *23rd International Conference on Rewriting Techniques and Applications (RTA'12)*, volume 15 of *LIPIcs*, pages 323–338. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2012.

26 Mitchell Wand. An algebraic formulation of the Chomsky hierarchy. In *Category Theory Applied to Computation and Control*, volume 25 of *LNCS*, pages 209–213. Springer, 1974.

27 Marek Zaionc. Word operation definable in the typed lambda-calculus. *Theor. Comput. Sci.*, 52:1–14, 1987. doi:10.1016/0304-3975(87)90077-6.

28 Marek Zaionc. On the "lambda"-definable tree operations. In *Algebraic Logic and Universal Algebra in Computer Science, Conference, Ames, Iowa, USA, June 1-4, 1988, Proceedings*, volume 425 of *Lecture Notes in Computer Science*, pages 279–292, 1990.