


Lambda-Definable Order-3 Tree Functions are Well-Quasi-Ordered

Kazuyuki Asada

Tohoku University, Sendai, Japan

asada@riec.tohoku.ac.jp

 <https://orcid.org/0000-0001-8782-2119>

Naoki Kobayashi

The University of Tokyo, Tokyo, Japan

koba@is.s.u-tokyo.ac.jp

Abstract

Asada and Kobayashi [ICALP 2017] conjectured a higher-order version of Kruskal's tree theorem, and proved a pumping lemma for higher-order languages modulo the conjecture. The conjecture has been proved up to order-2, which implies that Asada and Kobayashi's pumping lemma holds for order-2 tree languages, but remains open for order-3 or higher. In this paper, we prove a variation of the conjecture for order-3. This is sufficient for proving that a variation of the pumping lemma holds for order-3 tree languages (equivalently, for order-4 word languages).

2012 ACM Subject Classification Theory of computation → Lambda calculus

Keywords and phrases higher-order grammar, pumping lemma, Kruskal's tree theorem, well-quasi-ordering, simply-typed lambda calculus

Digital Object Identifier 10.4230/LIPIcs.FSTTCS.2018.14

Related Version A full version of the paper is available at <http://www.riec.tohoku.ac.jp/~asada/papers/fsttcs18.pdf>.

Acknowledgements We would like to thank anonymous referees for useful comments. This work was supported by JSPS Kakenhi 15H05706 and 18K11156.

1 Introduction

Kruskal's tree theorem [7] says that the homeomorphic embedding relation \preceq^{he} on finite trees is a well-quasi-ordering, i.e., for every infinite sequence of trees $\pi_0, \pi_1, \pi_2, \dots$, there exist $i < j$ such that $\pi_i \preceq^{\text{he}} \pi_j$. Here, $\pi \preceq^{\text{he}} \pi'$ means that there exists an embedding of the nodes of π to those of π' , preserving the labels and the ancestor/descendant relation. Asada and Kobayashi [2] considered a higher-order version $\preceq_{\kappa}^{\text{he}}$ of \preceq^{he} on simply-typed λ -terms of type κ , and conjectured that $\preceq_{\kappa}^{\text{he}}$ is also a well-quasi-ordering, for every simple type κ . Under the assumption that the conjecture (which we call AK-conjecture) is true, they proved a pumping lemma for higher-order languages (a la higher-order languages in Damm's IO hierarchy [3]), which says that for any order- k tree grammar that generates an infinite language L , there exists a strictly increasing infinite sequence $\pi_0 \prec^{\text{he}} \pi_1 \prec^{\text{he}} \pi_2 \prec^{\text{he}} \dots$ such that $\pi_i \in L$ and $|\pi_i| \leq \mathbf{exp}_k(ci + d)$, where \prec^{he} is the strict version of the homeomorphic embedding, c and d are constants that depend on the grammar, and $\mathbf{exp}_k(x)$ is defined by $\mathbf{exp}_0(x) = x$ and $\mathbf{exp}_{k+1}(x) = 2^{\mathbf{exp}_k(x)}$. The pumping lemma can be used to prove that a certain language does not belong to the class of order- k languages. They also proved that the conjecture is



© Kazuyuki Asada and Naoki Kobayashi;

licensed under Creative Commons License CC-BY

38th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS 2018).

Editors: Sumit Ganguly and Paritosh Pandya; Article No. 14; pp. 14:1–14:15



Leibniz International Proceedings in Informatics

Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

true up to order-2 types, and hence also the pumping lemma for order-2 tree languages and (by the correspondence between tree/word languages [1, 3]) order-3 word languages. The AK-conjecture is still open for order-3 or higher.

In the present paper, we consider a variation of the AK-conjecture (which we call nAK-conjecture), where the homeomorphic embedding relation is replaced by $\preceq^\#$, defined by $\pi_1 \preceq^\# \pi_2$ if and only if, for every tree constructor a , $\#_a(\pi_1) \leq \#_a(\pi_2)$; here $\#_a(\pi)$ denotes the number of occurrences of a in π . The correctness of the nAK-conjecture would imply the following variation of the pumping lemma: for any order- k tree grammar that generates an infinite language L , there exists a strictly increasing infinite sequence $\pi_0 \prec^\# \pi_1 \prec^\# \pi_2 \prec^\# \dots$ such that $\pi_i \in L$ and $|\pi_i| \leq \mathbf{exp}_k(ci + d)$. We prove that the nAK-conjecture is true for the order-3 case, i.e., that $\preceq_\kappa^\#$ (the logical relation on simply-typed λ -terms of type κ , obtained from $\preceq^\#$) is a well-quasi-ordering for any type κ of order up to 3. The variation of the pumping lemma above is thus obtained for order-3 tree languages and order-4 word languages. To our knowledge, pumping lemmas were known only for tree (word, resp.) languages of order up to 2 (3, resp.) [2].

Parikh
image

To prove the order-3 nAK-conjecture, we define a transformation $(\cdot)^\natural$ from order-3 λ -terms to order-2 numeric functions (that are also represented by λ -terms), and prove (i) the transformation reflects the quasi-orderings, i.e., $t_1 \preceq_\kappa^\# t_2$ if $t_1^\natural \preceq^\mathbb{N} t_2^\natural$ for a certain quasi-ordering $\preceq^\mathbb{N}$ on numeric functions, and (ii) $\preceq^\mathbb{N}$ is a well-quasi-ordering.

Related work. We are not aware of directly related work, besides our own previous work [2]. Our reduction from the well-quasi-orderedness of order-3 λ -terms to that of order-2 numeric functions relies on the inexpressiveness of simply-typed λ -terms as (higher-order) tree functions. Zaionc [11, 12, 13] studied the expressive power of simply-typed λ -terms. Pumping lemmas for higher-order languages have been known to be difficult. After Hayashi [5] proved a pumping lemma for indexed languages (i.e. order-2 word languages), it was only in 2017 that a pumping lemma for order-3 word languages was proved [2]. We have further improved the result to obtain a pumping lemma for order-4 word (or, order-3 tree) languages.

The rest of the paper is structured as follows. Section 2 introduces basic definitions. Section 3 explains the nAK-conjecture and the pumping lemma. Section 4 proves the nAK-conjecture up to order-3. Section 5 concludes the paper.

2 Preliminaries

We give basic definitions on λ -terms and quasi-orderings.

2.1 λ -terms and higher-order languages

► **Definition 1** (types and terms). The set of *simple types*, ranged over by κ , is given by: $\kappa ::= \mathbf{o} \mid \kappa_1 \rightarrow \kappa_2$. The order¹ of a simple type κ , written $\mathbf{order}(\kappa)$ is defined by $\mathbf{order}(\mathbf{o}) = 0$ and $\mathbf{order}(\kappa_1 \rightarrow \kappa_2) = \max(\mathbf{order}(\kappa_1) + 1, \mathbf{order}(\kappa_2))$. The type \mathbf{o} describes trees, and $\kappa_1 \rightarrow \kappa_2$ describes functions from κ_1 to κ_2 . A (*ranked*) *alphabet* Σ is a map from a finite set of constants (that represent tree constructors) to the set of natural numbers called *arities*. The set of λY^{nd} -terms, ranged over by s, t, u, v , is defined by:

$$t ::= x \mid a t_1 \cdots t_k \mid t_1 t_2 \mid \lambda x : \kappa. t \mid Y_\kappa t \mid t_1 \oplus t_2$$

¹ For clarity, we use the word *order* for this notion, and *ordering* for relations such as \leq , \preceq^{he} , etc.

Here, x, y, \dots ranges over variables, and a over $\text{dom}(\Sigma)$. The term $at_1 \cdots t_k$ (where we require $\Sigma(a) = k$) constructs a tree that has a as the root and (the values of) t_1, \dots, t_k as children. Y_κ and \oplus represent a fixed-point combinator and a non-deterministic choice, respectively. We often omit the type annotation and just write $\lambda x.t$ and $Y t$ for $\lambda x : \kappa.t$ and $Y_\kappa t$. A λY^{nd} -term is called: (i) a $\lambda^{\rightarrow, \text{nd}}$ -term if it does not contain Y ; (ii) a λ^{\rightarrow} -term if it contains neither Y nor \oplus ; and (iii) an *applicative term* if it contains none of λ -abstractions, Y , and \oplus . We often call a λ^{\rightarrow} -term just a *term*. As usual, we identify λY^{nd} -terms up to the α -equivalence, and implicitly apply α -conversions.

A *type environment* Γ is a sequence of type bindings of the form $x : \kappa$ such that Γ contains at most one binding for each variable x . A λY^{nd} -term t has type κ under Γ if $\Gamma \vdash_{\text{ST}} t : \kappa$ is derivable from the following typing rules.

$$\frac{}{\Gamma, x : \kappa, \Gamma' \vdash_{\text{ST}} x : \kappa} \quad \frac{\Sigma(a) = k \quad \Gamma \vdash_{\text{ST}} t_i : \circ \text{ (for each } i \in \{1, \dots, k\})}{\Gamma \vdash_{\text{ST}} at_1 \cdots t_k : \circ} \quad \frac{\Gamma \vdash_{\text{ST}} t : \kappa \rightarrow \kappa}{\Gamma \vdash_{\text{ST}} Y_\kappa t : \kappa}$$

$$\frac{\Gamma \vdash_{\text{ST}} t_1 : \kappa_2 \rightarrow \kappa \quad \Gamma \vdash_{\text{ST}} t_2 : \kappa_2}{\Gamma \vdash_{\text{ST}} t_1 t_2 : \kappa} \quad \frac{\Gamma, x : \kappa_1 \vdash_{\text{ST}} t : \kappa_2}{\Gamma \vdash_{\text{ST}} \lambda x : \kappa_1. t : \kappa_2} \quad \frac{\Gamma \vdash_{\text{ST}} t_1 : \circ \quad \Gamma \vdash_{\text{ST}} t_2 : \circ}{\Gamma \vdash_{\text{ST}} t_1 \oplus t_2 : \circ}$$

We consider below only well-typed λY^{nd} -terms. Note that given Γ and t , there exists at most one type κ such that $\Gamma \vdash_{\text{ST}} t : \kappa$. We call κ the type of t (with respect to Γ). We often omit “with respect to Γ ” if Γ is clear from context. Given a judgment $\Gamma \vdash t : \kappa$, we define $\lambda \Gamma.t$ by: $\lambda \emptyset.t := t$ and $\lambda(\Gamma, x : \kappa').t := \lambda \Gamma. \lambda x.t$. Also we define $\Gamma \rightarrow \kappa$ by: $\emptyset \rightarrow \kappa := \kappa$ and $(\Gamma, x : \kappa') \rightarrow \kappa := \Gamma \rightarrow (\kappa' \rightarrow \kappa)$; thus we have $\vdash \lambda \Gamma.t : \Gamma \rightarrow \kappa$ if $\Gamma \vdash t : \kappa$. Given an alphabet Σ , we write Λ^Σ for the set of λ^{\rightarrow} -terms whose constants are taken from Σ . Also we define $\Lambda_{\Gamma, \kappa}^\Sigma := \{t \in \Lambda^\Sigma \mid \Gamma \vdash t : \kappa\}$ and $\Lambda_{\emptyset, \kappa}^\Sigma := \Lambda_{\emptyset, \kappa}^\Sigma$.

For a λY^{nd} -term t with a type environment Γ , the (*internal*) *order* of t (with respect to Γ), written $\text{order}_\Gamma(t)$, is the largest order of the types of subterms of $\lambda \Gamma.t$, and the *external order* of t (with respect to Γ), written $\text{eorder}_\Gamma(t)$, is the order of the type of t with respect to Γ . We often omit Γ when it is clear from context. For example, for $t = (\lambda x : \circ.x)\mathbf{e}$, $\text{order}_\emptyset(t) = 1$ and $\text{eorder}_\emptyset(t) = 0$. We define the *size* $|t|$ of a λY^{nd} -term t by: $|x| := 1$, $|at_1 \cdots t_k| := 1 + |t_1| + \cdots + |t_k|$, $|st| := |s| + |t| + 1$, $|\lambda x.t| := |t| + 1$, $|Y_\kappa t| := |t| + 1$ and $|s \oplus t| := |s| + |t| + 1$. We call a λY^{nd} -term t *ground* (with respect to Γ) if $\Gamma \vdash_{\text{ST}} t : \circ$. We call t a (finite, Σ -ranked) *tree* if t is a ground closed applicative term (consisting of only constants). We write \mathbf{Tree}_Σ for the set of Σ -ranked trees, and use the meta-variable π for a tree. We often write $\vec{\tau}$ to denote a sequence (possibly with a condition on the range of the sequence in the superscript). For example, $\vec{t}_i^{i \leq m}$ denotes the sequence t_1, \dots, t_m of terms, and $\vec{[t_i/x_i]^{i \leq m}}$ denotes the substitution $[t_1/x_1, \dots, t_m/x_m]$.

We sometimes identify a ranked alphabet $\Sigma = \{a_1 \mapsto r_1, \dots, a_k \mapsto r_k\}$ with the first-order environment $\Sigma = \{a_1 : \circ^{r_1} \rightarrow \circ, \dots, a_k : \circ^{r_k} \rightarrow \circ\}$ (assuming an arbitrary fixed linear ordering on Σ).

► **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 λY^{nd} -terms is defined by:

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

where $t[t'/x]$ 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 λY^{nd} -term t such that

$t \not\rightarrow t'$ for any t' . For a ground closed λY^{nd} -term t , we define the *tree language* $\mathcal{L}(t)$ generated by t by $\mathcal{L}(t) := \{\pi \mid t \longrightarrow^* \pi\}$. For a ground closed λ^{\rightarrow} -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* .

In the previous paper [2] we stated the pumping lemma for the notion of a *higher-order grammar*; in this paper, following [8, 9], we use only the formalism by λY^{nd} -terms for simplicity. Since there exist well-known order-preserving and language-preserving transformations between higher-order grammars and ground closed λY^{nd} -terms, we obtain corresponding results on higher-order grammars immediately.

The notion of a word can be seen as a special case of that of a tree:

► **Definition 3** (word alphabet). We call a ranked alphabet Σ a *word alphabet* if it has a special nullary constant \mathbf{e} and all the other constants have arity 1. For a tree $\pi = a_1(\cdots(a_n \mathbf{e})\cdots)$ of a word alphabet, we define $\mathbf{word}(\pi) := a_1 \cdots a_n$, and we define \mathbf{utree} as the inverse function of \mathbf{word} , i.e., $\mathbf{utree}(a_1 \cdots a_n) := a_1(\cdots(a_n \mathbf{e})\cdots)$. The *word language* generated by a ground closed λY^{nd} -term t over a word alphabet, written $\mathcal{L}_w(t)$, is defined as $\{\mathbf{word}(\pi) \mid \pi \in \mathcal{L}(t)\}$.

A tree language (word language, resp.) over an alphabet (word alphabet, resp.) Σ is called *order- n* if it is generated by some order- n ground closed λY^{nd} -term of Σ ; 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 [10].

2.2 Some quasi-orderings and their logical relation extension

► **Definition 4** ((well-)quasi-ordering). A *quasi-ordering* (a.k.a. preorder) on a set A is a binary relation on A that is reflexive and transitive. A *well-quasi-ordering* (*wqo* for short) on a set S is a quasi-ordering \leq on S such that for any infinite sequence $(s_i)_i$ of elements in S there exist j and k such that $j < k$ and $s_j \leq s_k$.

As a general notation, for a quasi-ordering denoted by \preceq , we write \approx for the induced equivalence relation (i.e., $x \approx y$ if $x \preceq y$ and $y \preceq x$), and write \prec for the strict version (i.e., $x \prec y$ if $x \preceq y$ and $y \not\preceq x$). Also, for a quasi-ordering denoted by \leq , we write \sim for the induced equivalence relation and $<$ for the strict version. We apply these conventions also to notations with superscript/subscript such as \preceq^a , \preceq_b , \preceq_b^a , \leq^a , \leq_b , and \leq_b^a . Further, for any quasi-ordering on the set of trees of a word alphabet, we use the same notation also for the quasi-ordering on the set of words induced through \mathbf{utree} .

► **Definition 5** (logical relation extension). Let Σ be a ranked alphabet. We call \leq a *base quasi-ordering* (with respect to Σ) if \leq is a quasi-ordering on the set Λ_o^Σ modulo $\beta\eta$ -equivalence and every constant in Σ is monotonic on \leq . We define the *logical relation extension* of \leq as the family $(\leq_\kappa)_\kappa$ of relations \leq_κ on the set Λ_κ^Σ modulo $\beta\eta$ -equivalence indexed by simple types κ where \leq_κ 's are defined by induction on κ as follows:

$$\begin{aligned} t_1 \leq_o t_2 & \quad \text{if} \quad t_1 \leq t_2 \\ t_1 \leq_{\kappa \rightarrow \kappa'} t_2 & \quad \text{if} \quad \text{for any } t'_1, t'_2, \quad t'_1 \leq_\kappa t'_2 \implies t_1 t'_1 \leq_{\kappa'} t_2 t'_2. \end{aligned}$$

Furthermore we extend the relation to open terms: for $t_1, t_2 \in \Lambda_{\Gamma, \kappa}^\Sigma$, we define $t_1 \leq_{\Gamma, \kappa} t_2$ if $\lambda\Gamma.t_1 \leq_{\Gamma \rightarrow \kappa} \lambda\Gamma.t_2$. We omit the subscripts of \leq_κ and $\leq_{\Gamma, \kappa}$ if there is no confusion.

The next lemma follows immediately from the basic lemma (a.k.a. the abstraction theorem) of logical relations (see the full version for details).

► **Lemma 6.** *Let \leq be a base quasi-ordering. Each component \leq_κ of the logical relation extension of \leq is a quasi-ordering. Further, \leq_κ is the point-wise quasi-ordering:*

$$t_1 \leq_{\kappa \rightarrow \kappa'} t_2 \quad \text{if and only if} \quad \text{for any } t' \in \Lambda_\kappa^\Sigma, \quad t_1 t' \leq_{\kappa'} t_2 t'.$$

Every quasi-ordering for higher-order terms used in this paper is a logical relation extension (of some base quasi-ordering). The next ordering is used in the previous paper [2].

► **Definition 7** (homeomorphic embedding). Let Σ be a ranked alphabet. The *homeomorphic embedding* ordering $\preceq^{\text{he}, \Sigma}$ between Σ -ranked trees² is inductively defined by the following rules:

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

We extend the above ordering to a base ordering by: $t_1 \preceq^{\text{he}, \Sigma} t_2$ if $\mathcal{T}(t_1) \preceq^{\text{he}, \Sigma} \mathcal{T}(t_2)$.

For example, $\text{br a b} \preceq^{\text{he}} \text{br (br a c) b}$. The homeomorphic embedding on words is nothing but the (scattered) subsequence ordering. The following is a fundamental result on the homeomorphic embedding:

► **Proposition 8** (Kruskal's tree theorem [7]). *For any (finite) ranked alphabet Σ , the homeomorphic embedding \preceq^{he} on Σ -ranked trees is a well-quasi-ordering.*

Also, we often use the Dickson's theorem [6] which says that the product quasi-ordering (component-wise quasi-ordering) of a finite number of wqo's is a wqo.

The next is the quasi-ordering that is used in the theorems in this paper.

► **Definition 9** (occurrence-number quasi-ordering). Let Σ be a ranked alphabet. For $a \in \Sigma$ and a Σ -tree π , we define $\#_a(\pi)$ as the number of occurrences of a in π , and extend this to a ground closed λ^\rightarrow -term t by $\#_a(t) := \#_a(\mathcal{T}(t))$. Then we define a base quasi-ordering $\preceq^{\#, \Sigma, a}$ by:

$$t_1 \preceq^{\#, \Sigma, a} t_2 \quad \text{if} \quad \#_a(t_1) \leq \#_a(t_2).$$

Also we define a base quasi-ordering $\preceq^{\#, \Sigma}$ by:

$$t_1 \preceq^{\#, \Sigma} t_2 \quad \text{if} \quad \text{for every } a \in \Sigma, \quad t_1 \preceq^{\#, \Sigma, a} t_2.$$

Note that $\pi \preceq^{\text{he}} \pi'$ implies $\pi \preceq^{\#, \Sigma} \pi'$, shown by induction on the rule of \preceq^{he} ; and further $\pi \preceq_\kappa^{\text{he}} \pi'$ implies $\pi \preceq_\kappa^{\#, \Sigma} \pi'$ for any κ since $\preceq_\kappa^{\text{he}}$ and $\preceq_\kappa^{\#, \Sigma}$ are point-wise quasi-ordering. Also note that $\preceq_\kappa^{\#, \Sigma} = \bigcap_{a \in \Sigma} (\preceq_\kappa^{\#, \Sigma, a})$ for any κ .

The next quasi-ordering is used just in proofs. We write $\Sigma_{\mathbb{N}}$ for the ranked alphabet $\{0 \mapsto 0, 1 \mapsto 0, + \mapsto 2, \times \mapsto 2\}$; we write $+tt'$ as $t + t'$ and $\times tt'$ as $t \times t'$. We define a set-theoretical denotational interpretation $\llbracket - \rrbracket$ of $\Lambda^{\Sigma_{\mathbb{N}}}$ by: $\llbracket 0 \rrbracket := \mathbb{N}$, $\llbracket \kappa \rightarrow \kappa' \rrbracket$ is the set of functions from $\llbracket \kappa \rrbracket$ to $\llbracket \kappa' \rrbracket$, $\llbracket 0 \rrbracket := 0$, $\llbracket 1 \rrbracket := 1$, $\llbracket + \rrbracket(n)(m) := n + m$, and $\llbracket \times \rrbracket(n)(m) := n \times m$. For $t_1, t_2 \in \Lambda_{\Gamma, \kappa}^{\Sigma_{\mathbb{N}}}$, we write $t_1 =_{\Gamma, \kappa}^{\llbracket \cdot \rrbracket} t_2$ (or $t_1 =^{\llbracket \cdot \rrbracket} t_2$) if $\llbracket t_1 \rrbracket = \llbracket t_2 \rrbracket$.

► **Definition 10** (natural number quasi-ordering). We define a base quasi-ordering $\preceq^{\mathbb{N}}$ on the set $\Lambda_{\circ}^{\Sigma_{\mathbb{N}}}$ by:

$$t_1 \preceq^{\mathbb{N}} t_2 \quad \text{if} \quad \llbracket t_1 \rrbracket \leq \llbracket t_2 \rrbracket.$$

² In the usual definition, a quasi-ordering on labels (tree constructors) is assumed. Here we fix the quasi-order on labels to the identity relation.

3 Numeric Pumping Lemma for Higher-order Tree Languages

Here we explain the nAK-conjecture and the pumping lemma for higher-order tree languages with respect to $\preceq^{\#, \Sigma}$.

► **Conjecture 11** (nAK-conjecture). *For any Σ and κ , $\preceq^{\#, \Sigma}_\kappa$ is a well quasi-ordering.*

Our main theorem (Theorem 14) is to show the above conjecture for κ of order up to 3. The above conjecture (and Theorem 14) can be used for the following pumping lemma:

► **Theorem 12** (pumping lemma). *Assume that Conjecture 11 holds. Then, for any order- n ground closed λY^{nd} -term t of a ranked alphabet Σ such that $\mathcal{L}(t)$ is infinite, there exist an infinite sequence of trees $\pi_0, \pi_1, \pi_2, \dots \in \mathcal{L}(t)$, and constants c, d such that:*

- (i) $\pi_0 \prec^{\#, \Sigma} \pi_1 \prec^{\#, \Sigma} \pi_2 \prec^{\#, \Sigma} \dots$, and
- (ii) $|\pi_i| \leq \exp_n(ci + d)$ for each $i \geq 0$.

Furthermore, we can drop the assumption on Conjecture 11 when $n \leq 3$.

The proof of the above theorem is obtained as a simple modification of the proof of the pumping lemma in [2]: see the full version.

► **Remark.** The theorem we prove in the full version is actually slightly stronger than Theorem 12 above, in the following three points (see the full version for details):

- (i) As in [2], we relax the assumption of nAK conjecture, so that $\preceq^{\#, \Sigma}_\kappa$ need not be the logical relation; any higher-order extension of the base quasi-ordering that is closed under application suffices.
- (ii) As in [2], we use actually a weaker conjecture, called the *periodicity*, which requires that, for any $\vdash_{\text{ST}} t : \kappa \rightarrow \kappa$ and $\vdash_{\text{ST}} s : \kappa$, there exist $i, j > 0$ such that $t^i s \preceq^{\#, \Sigma}_\kappa t^{i+j} s \preceq^{\#, \Sigma}_\kappa t^{i+2j} s \preceq^{\#, \Sigma}_\kappa \dots$.
- (iii) Whilst Theorem 12 states a pumping lemma on $\preceq^{\#, \Sigma}$, the generalized theorem states a pumping lemma on arbitrary base quasi-ordering with certain conditions, which includes $\preceq^{\#, \Sigma}$ and \preceq^{he} as instances.

By the correspondence between order- n tree grammars and order- $(n+1)$ word grammars [3, 1], we also have:

► **Corollary 13** (pumping lemma for word languages). *Assume that Conjecture 11 holds. Then, for any order- n ground closed λY^{nd} -term t of a word alphabet Σ (where $n \geq 1$) such that $\mathcal{L}_w(t)$ is infinite, there exist an infinite sequence of words $w_0, w_1, w_2, \dots \in \mathcal{L}_w(t)$, and constants c, d such that:*

- (i) $w_0 \prec^{\#, \Sigma} w_1 \prec^{\#, \Sigma} w_2 \prec^{\#, \Sigma} \dots$, and
- (ii) $|w_i| \leq \exp_{n-1}(ci + d)$ for each $i \geq 0$.

Furthermore, we can drop the assumption on Conjecture 11 when $n \leq 4$.

4 Numeric Version of Order-3 Kruskal's Tree Theorem

Here we prove the main theorem (Theorem 14 below), which states that the nAK-conjecture (Conjecture 11) holds for order-3 types. In this whole section, by a *term*, we mean a λ^\rightarrow -term, and we never consider a fixed-point combinator nor non-determinism.

4.1 Main theorem

► **Theorem 14.** *For any alphabet Σ and any type κ of order up to 3, $\preceq_{\kappa}^{\#, \Sigma}$ on $\Lambda_{\kappa}^{\Sigma}$ is a wqo.*

The theorem above is obtained as a corollary of the following lemma.

► **Lemma 15.** *For any alphabet Σ , any $a \in \Sigma$, and any order-2 type environment Γ (i.e., a type environment whose codomain consists of types of order up to 2), the quasi-ordering $\preceq_{\Gamma, \circ}^{\#, \Sigma, a}$ on $\Lambda_{\Gamma, \circ}^{\emptyset}$ is a wqo.*

Proof sketch of Theorem 14.

- For Theorem 14, it is sufficient that $\preceq_{\kappa}^{\#, \Sigma, a}$ on $\Lambda_{\kappa}^{\Sigma}$ is a wqo for every $a \in \Sigma$ and κ with $\text{order}(\kappa) \leq 3$, because $\preceq_{\kappa}^{\#, \Sigma} = \bigcap_{a \in \Sigma} (\preceq_{\kappa}^{\#, \Sigma, a})$ and well-quasi-orderings are closed under finite intersection.
- For $\preceq_{\kappa}^{\#, \Sigma, a}$ to be a wqo for every order-3 type κ , it is sufficient that the restriction of $\preceq_{\kappa}^{\#, \Sigma, a}$ to $\Lambda_{\kappa}^{\emptyset}$ (i.e. $\preceq_{\kappa}^{\#, \Sigma, a} \cap (\Lambda_{\kappa}^{\emptyset} \times \Lambda_{\kappa}^{\emptyset})$) is a wqo for every order-3 type κ , because $t_1 \preceq_{\kappa}^{\#, \Sigma, a} t_2$ holds if $\lambda \Sigma. t_1 (\preceq_{\Sigma \rightarrow \kappa}^{\#, \Sigma, a} \cap (\Lambda_{\Sigma \rightarrow \kappa}^{\emptyset} \times \Lambda_{\Sigma \rightarrow \kappa}^{\emptyset})) \lambda \Sigma. t_2$, and $\text{order}(\Sigma \rightarrow \kappa) \leq 3$.
- For $\preceq_{\kappa}^{\#, \Sigma, a} \cap (\Lambda_{\kappa}^{\emptyset} \times \Lambda_{\kappa}^{\emptyset})$ to be a wqo, Lemma 15 is sufficient, because $t_1 (\preceq_{\kappa}^{\#, \Sigma, a} \cap (\Lambda_{\kappa}^{\emptyset} \times \Lambda_{\kappa}^{\emptyset})) t_2$ holds if $t_1 z_1 \cdots z_k \preceq_{\Gamma, \circ}^{\#, \Sigma, a} t_2 z_1 \cdots z_k$, where $\kappa = \kappa_1 \rightarrow \cdots \rightarrow \kappa_k \rightarrow \circ$ and $\Gamma = z_1 : \kappa_1, \dots, z_k : \kappa_k$.

See the full version for details. ◀

Henceforth, we fix arbitrary $a_{\text{fix}} \in \Sigma$, and show Lemma 15 for $a = a_{\text{fix}}$. We prove this lemma in two steps: First we give a transformation $(\cdot)^{\natural}$ from order-3 terms in $\Lambda_{\Gamma, \circ}^{\emptyset}$ (and their type environment Γ) to order-2 terms in $\Lambda_{\Gamma^{\natural}, \circ}^{\Sigma_{\text{fix}}}$ (and to Γ^{\natural}) so that it reflects quasi-orderings: $t^{\natural} \preceq_{\Gamma^{\natural}, \circ}^{\Sigma_{\text{fix}}} t'^{\natural}$ implies $t \preceq_{\Gamma, \circ}^{\#, \Sigma, a_{\text{fix}}} t'$ (Lemma 18). Then we show that $\preceq_{\Gamma^{\natural}, \circ}^{\Sigma_{\text{fix}}}$ on $\Lambda_{\Gamma^{\natural}, \circ}^{\Sigma_{\text{fix}}}$ is a wqo (Lemma 19). From these two results, Lemma 15 follows immediately.

4.2 Transformation from order-3 terms to order-2 terms

The key observation behind the transformation $(\cdot)^{\natural}$ is as follows. Let s be a closed term of type $\circ^m \rightarrow \circ$ and t_1, \dots, t_m be closed terms of type \circ . Then, we have:

$$\#_a(s t_1 \cdots t_m) = c_1 \times \#_a(t_1) + \cdots + c_m \times \#_a(t_m) + d$$

for some numbers c_1, \dots, c_m, d that do not depend on t_1, \dots, t_m . This is because the order-1 function s representable as a λ^{\rightarrow} -term can copy only arguments, and the number of copies cannot depend on the arguments. Thus, if we are interested only in the number of occurrences of a constant, information about an order-1 function can be represented by a tuple (c_1, \dots, c_m, d) of numbers (order-0 values, in other words). By lifting this representation to order-3 terms in $\Lambda_{\Gamma, \circ}^{\emptyset}$, we obtain order-2 terms in $\Lambda_{\Gamma^{\natural}, \circ}^{\Sigma_{\text{fix}}}$.

The actual transformation is non-trivial. Let us first fix $\Gamma = \varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_{\ell} : \circ^{q_{\ell}} \rightarrow \circ$. Here, φ_i 's are order-2 variables and f_j 's are variables of order up to 1. Every element of $\Lambda_{\Gamma, \circ}^{\emptyset}$ can be normalized to a term generated by the following syntax (which we call an *order-3 normal form*):

$$t ::= y \mid f_j \mid t_1 t_2 \mid \varphi_i t_1 \cdots t_k \mid \lambda y. t.$$

Here, y is a local variable of order 0. We require that the order of $\varphi t_1 \cdots t_k$ is at most 1. For example, $\varphi : (\circ \rightarrow \circ) \rightarrow \circ \rightarrow \circ \rightarrow \circ$, $f : \circ \rightarrow \circ \rightarrow \circ$, $x : \circ \vdash \lambda y : \circ. \varphi(f x)((\lambda y' : \circ. f y' y') y) : \circ \rightarrow \circ \rightarrow \circ$ is an order-3 normal form. It can be checked by induction that for any order-3 normal form t , $\text{eorder}_{\Gamma}(t) \leq 1$ (with a suitable environment Γ). Since any

long $\beta\eta$ -normal form in $\Lambda_{\Gamma, \circ}^\emptyset$ with $\text{order}(\Gamma \rightarrow \circ) = 3$ is an order-3 normal form, considering only order-3 normal forms does not lose generality. In the rest of this section, we use the meta-variable t for order-3 normal forms.

We now define the transformation for order-3 normal forms. Given a term $t_0 \in \Lambda_{\Gamma, \circ}^\emptyset$, we transform the term in a compositional manner, by transforming each subterm t typed by:

$$\varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_\ell : \circ^{q_\ell} \rightarrow \circ; y_1 : \circ, \dots, y_n : \circ \vdash t : \circ^r \rightarrow \circ$$

to a term e with some suitable type environment. Here, y_1, \dots, y_n are order-0 variables that are bound inside t_0 (rather than t), $\text{order}(\kappa_i) = 2$ for $i \leq m$, and $q_i \geq 0$ for $i \leq \ell$. We call f_i and φ_i *external variables* and y_i an *internal variable*. Note that an external variable f_i can be order-0.

We first explain how variables and environments are transformed.

- The variables y_1, \dots, y_n will just disappear after the transformation.
- For each order-1 variable f_i of type $\circ^{q_i} \rightarrow \circ$, we prepare a tuple of variables $(c_{f_i,1}, \dots, c_{f_i,q_i}, d_{f_i})$. Each $c_{f_i,j}$ expresses how often f_i copies the j -th argument, and d_{f_i} expresses how often a_{fix} occurs in the value of f_i , so that the number of a_{fix} in $f_i t_1, \dots, t_{q_i}$ can be represented by $c_{f_i,1} \times \#_{a_{\text{fix}}}(t_1) + \dots + c_{f_i,q_i} \times \#_{a_{\text{fix}}}(t_{q_i}) + d_{f_i}$ (recall the observation given at the beginning of this subsection).
- For each order-2 variable φ_i of type $\kappa_i = (\circ^{q_1} \rightarrow \circ) \rightarrow \dots \rightarrow (\circ^{q_k} \rightarrow \circ) \rightarrow (\circ^q \rightarrow \circ)$ (where $q_k > 0$), we prepare a tuple of order-1 variables $(g_{\varphi_i,1}, \dots, g_{\varphi_i,q}, h_{\varphi_i}, \hat{h}_{\varphi_i})$. Basically, $g_{\varphi_i,j}$ and h_{φ_i} are analogous to $c_{f_i,j}$ and d_{f_i} , respectively. Given order-1 functions t_1, \dots, t_k whose values are $\vec{u}_1, \dots, \vec{u}_k$ (where each \vec{u}_k is a tuple of size $q_k + 1$), for each $j \leq q$, the function $\varphi_i t_1 \dots t_k$ copies the j -th order-0 argument $g_{\varphi_i,j}(\vec{u}_1, \dots, \vec{u}_k)$ times, and creates $h_{\varphi_i}(\vec{u}_1, \dots, \vec{u}_k)$ copies of the constant a_{fix} . The other function variable \hat{h}_{φ_i} is similar to h_{φ_i} but used for counting an internal variable y_j rather than a_{fix} .

For a type environment

$$\Gamma = \varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_\ell : \circ^{q_\ell} \rightarrow \circ$$

where $\kappa_i = (\circ^{q_1^i} \rightarrow \circ) \rightarrow \dots \rightarrow (\circ^{q_{k_i}^i} \rightarrow \circ) \rightarrow (\circ^{q^i} \rightarrow \circ)$ ($q_{k_i}^i > 0$, $i = 1, \dots, k$), we define:

$$\Gamma^\sharp := \frac{\frac{\frac{}{g_{\varphi_i,j} \rightarrow j \leq q^i}, h_{\varphi_i}, \hat{h}_{\varphi_i} : \circ^{q_1^i+1} \rightarrow \dots \rightarrow \circ^{q_{k_i}^i+1} \rightarrow \circ}{\rightarrow j \leq m}}{\rightarrow j \leq q^i}, \frac{}{c_{f_i,j} \rightarrow j \leq q_i}, d_{f_i} : \circ$$

We now define the transformation of terms. A term t such that

$$\varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_\ell : \circ^{q_\ell} \rightarrow \circ; y_1 : \circ, \dots, y_n : \circ \vdash t : \circ^r \rightarrow \circ$$

is transformed to a tuple $(v_1, \dots, v_n; w_1, \dots, w_r; e)$, using the transformation relation

$$\varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_\ell : \circ^{q_\ell} \rightarrow \circ; y_1 : \circ, \dots, y_n : \circ \vdash t \triangleright (v_1, \dots, v_n; w_1, \dots, w_r; e)$$

defined below. Here, each component is constructed from variables $c_{f_i,j}, d_{f_i}, g_{\varphi_i,j}, h_{\varphi_i}, \hat{h}_{\varphi_i}$ above and $\times, +, 0, 1$. The output of the transformation consists of three parts, separated by semicolons: a (possibly empty) sequence v_1, \dots, v_n , a (possibly empty) sequence w_1, \dots, w_r , and a single element e . The term v_j represents how often y_j is copied, w_j represents how often the j -th argument of t is copied, and e represents how often the constant a_{fix} is copied. The terms v_j and w_j are auxiliary ones for this transformation, and e plays the role of t^\sharp explained in Section 4.1.

The transformation relation is defined by the following rules, where $\Gamma = \varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \mathbf{o}^{q_1} \rightarrow \mathbf{o}, \dots, f_\ell : \mathbf{o}^{q_\ell} \rightarrow \mathbf{o}$ is fixed.

$$\frac{}{\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash y_j \triangleright (\underbrace{0, \dots, 0}_{j-1}, 1, \underbrace{0, \dots, 0}_{n-j}; 0)} \quad (\text{IVAR})$$

$$\frac{}{\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash f_i \triangleright (\underbrace{0, \dots, 0}_n; c_{f_i,1}, \dots, c_{f_i,q_i}; d_{f_i})} \quad (\text{VAR})$$

$$\frac{\begin{array}{c} \Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash t_1 \triangleright (v_1, \dots, v_n; w_1, \dots, w_r; e) \quad r \geq 1 \\ \Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash t_2 \triangleright (v'_1, \dots, v'_n; e') \end{array}}{\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash t_1 t_2 \triangleright (v_1 + w_1 v'_1, \dots, v_n + w_1 v'_n; w_2, \dots, w_r; e + w_1 e')} \quad (\text{APP0})$$

$$\frac{\begin{array}{c} \Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash t_j \triangleright (\vec{w}_j; \vec{w}_j; e_j) \quad \vec{u}_j = (\vec{w}_j; e_j) \quad (\text{for each } j \in \{1, \dots, k\}) \\ \vec{u}'_{j,j'} = (\vec{w}_j; v_{j,j'}) \quad (\text{for each } j \in \{1, \dots, k\} \text{ and } j' \in \{1, \dots, n\}) \\ k \geq 1 \text{ and the type of } t_k \text{ is order-1} \end{array}}{\begin{array}{c} \Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash \varphi_i t_1 \dots t_k \triangleright \\ (\hat{h}_{\varphi_i}(\vec{u}'_{1,1}, \dots, \vec{u}'_{k,1}) \dots, \hat{h}_{\varphi_i}(\vec{u}'_{1,n}, \dots, \vec{u}'_{k,n}); \\ g_{\varphi_i,1}(\vec{u}_1, \dots, \vec{u}_k), \dots, g_{\varphi_i,q_i}(\vec{u}_1, \dots, \vec{u}_k); h_{\varphi_i}(\vec{u}_1, \dots, \vec{u}_k)) \end{array}} \quad (\text{APP1})$$

$$\frac{\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o}, y_{n+1} : \mathbf{o} \vdash t \triangleright (v_1, \dots, v_n, v_{n+1}; w_1, \dots, w_r; e)}{\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash \lambda y_{n+1}. t \triangleright (v_1, \dots, v_n; v_{n+1}, w_1, \dots, w_r; e)} \quad (\text{LAM})$$

Rules (IVAR) (for internal variables of type \mathbf{o}) (VAR) (for order-1 variables), and (LAM) should be obvious from the intuition on the tuple and the translation of an environment. Rules (APP0) and (APP1) are for applications of order-1 and order-2 functions respectively. (Note however that in (APP0), t_1 itself may be an application of order-2 function, of the form $\varphi t_{1,1} \dots t_{1,k}$.) In (APP0), note that $t_1 t_2$ creates w_1 copies of (the value of) t_2 , so that the number of copies of y_i can be calculated by $v_i + w_1 v'_i$, where v_i and v'_i are the numbers of copies created by t_1 and t_2 respectively. Rule (APP1) is based on the intuition explained above about the translation of order-2 variables. Note that the same function \hat{h}_{φ_i} is used for counting y_1, \dots, y_n ; this is because φ_i does not know y_j (in other words, φ_i cannot be instantiated to a term containing y_j as a free variable), so that the information for counting y_j can only be passed through arguments $\vec{u}'_{j,j'}$.

It should be clear that if $\Gamma; y_1 : \mathbf{o}, \dots, y_n : \mathbf{o} \vdash t \triangleright (v_1, \dots, v_n; w_1, \dots, w_r; e)$ then $v_j, w_{j'}, e \in \Lambda_{\Gamma^\sharp, \mathbf{o}}^{\Sigma_N}$ and the order of $\Gamma^\sharp \rightarrow \mathbf{o}$ is no greater than 2.

► **Example 16.** Let $\Gamma = \varphi : (\mathbf{o} \rightarrow \mathbf{o}) \rightarrow \mathbf{o} \rightarrow \mathbf{o}, f : \mathbf{o} \rightarrow \mathbf{o}$. Then, we have

$$\Gamma^\sharp = g_{\varphi,1}, h_{\varphi}, \hat{h}_{\varphi} : \mathbf{o}^2 \rightarrow \mathbf{o}, c_{f,1}, d_f : \mathbf{o}$$

and $t := \lambda y. \varphi(\varphi f) y$ is transformed to

$$t^\sharp = h_{\varphi}(g_{\varphi,1}(c_{f,1}, d_f), h_{\varphi}(c_{f,1}, d_f)) + g_{\varphi,1}(g_{\varphi,1}(c_{f,1}, d_f), h_{\varphi}(c_{f,1}, d_f)) \times 0$$

by the following derivation:

$$\begin{array}{c}
\frac{}{\Gamma; y : \circ \vdash f \triangleright (0; c_{f,1}; d_f)} \text{ (VAR)} \\
\frac{}{\Gamma; y : \circ \vdash \varphi f \triangleright (\hat{h}_\varphi(c_{f,1}, 0); g_{\varphi,1}(c_{f,1}, d_f); h_\varphi(c_{f,1}, d_f))} \text{ (APP1)} \\
\frac{}{\Gamma; y : \circ \vdash \varphi(\varphi f) \triangleright (\hat{h}_\varphi(\vec{u}'); g_{\varphi,1}(\vec{u}); h_\varphi(\vec{u}))} \text{ (APP1)} \quad \frac{}{\Gamma; y : \circ \vdash y \triangleright (1; 0)} \text{ (IVAR)} \\
\frac{}{\Gamma; y : \circ \vdash \varphi(\varphi f) y \triangleright (\hat{h}_\varphi(\vec{u}') + g_{\varphi,1}(\vec{u}) \times 1; h_\varphi(\vec{u}) + g_{\varphi,1}(\vec{u}) \times 0)} \text{ (APP0)} \\
\frac{}{\Gamma; \vdash \lambda y. \varphi(\varphi f) y \triangleright (\hat{h}_\varphi(\vec{u}') + g_{\varphi,1}(\vec{u}) \times 1; h_\varphi(\vec{u}) + g_{\varphi,1}(\vec{u}) \times 0)} \text{ (LAM)}
\end{array}$$

where $\vec{u} = g_{\varphi,1}(c_{f,1}, d_f), h_\varphi(c_{f,1}, d_f)$ and $\vec{u}' = g_{\varphi,1}(c_{f,1}, d_f), \hat{h}_\varphi(c_{f,1}, 0)$. The terms in the bottom line of the derivation, $\hat{h}_\varphi(\vec{u}') + g_{\varphi,1}(\vec{u}) \times 1$ and $t^\sharp = h_\varphi(\vec{u}) + g_{\varphi,1}(\vec{u}) \times 0$, have type \circ under the environment Γ^\sharp , and $\mathbf{eorder}(\lambda \Gamma^\sharp. t^\sharp) = \mathbf{order}(\Gamma^\sharp \rightarrow \circ) = 2$.

The next example is a slightly modified one involving an external variable $x : \circ$ instead of the internal variable $y : \circ$. We have

$$(\Gamma, x : \circ)^\sharp = \Gamma^\sharp, d_x : \circ$$

and $t' := \varphi(\varphi f) x$ is transformed to

$$t'^\sharp = h_\varphi(g_{\varphi,1}(c_{f,1}, d_f), h_\varphi(c_{f,1}, d_f)) + g_{\varphi,1}(g_{\varphi,1}(c_{f,1}, d_f), h_\varphi(c_{f,1}, d_f)) \times d_x$$

by the following derivation:

$$\begin{array}{c}
\frac{}{\Gamma, x : \circ; \vdash f \triangleright (0; c_{f,1}; d_f)} \text{ (VAR)} \\
\frac{}{\Gamma, x : \circ; \vdash \varphi f \triangleright (\hat{h}_\varphi(c_{f,1}, 0); g_{\varphi,1}(c_{f,1}, d_f); h_\varphi(c_{f,1}, d_f))} \text{ (APP1)} \\
\frac{}{\Gamma, x : \circ; \vdash \varphi(\varphi f) \triangleright (\hat{h}_\varphi(\vec{u}'); g_{\varphi,1}(\vec{u}); h_\varphi(\vec{u}))} \text{ (APP1)} \quad \frac{}{\Gamma, x : \circ; \vdash x \triangleright (0; d_x)} \text{ (VAR)} \\
\frac{}{\Gamma, x : \circ; \vdash \varphi(\varphi f) x \triangleright (\hat{h}_\varphi(\vec{u}') + g_{\varphi,1}(\vec{u}) \times 0; h_\varphi(\vec{u}) + g_{\varphi,1}(\vec{u}) \times d_x)} \text{ (APP0)}
\end{array}$$

where \vec{u} and \vec{u}' are the same as above. \blacktriangleleft

Lemma 17 below says that the transformation preserves the meaning of ground terms. Here we regard constants in Σ as variables of up to order 1, and we define a substitution $\theta_\Sigma^{a_{\text{fix}}}$ by:

$$\theta_\Sigma^{a_{\text{fix}}} := \left[\frac{}{1/c_{a,i}} \xrightarrow{a \in \Sigma, i \leq \mathbf{ar}(a)}, \frac{}{1/d_{a_{\text{fix}}}}, \frac{}{0/d_a} \xrightarrow{a \in \Sigma \setminus \{a_{\text{fix}}\}} \right].$$

(Recall that $a_{\text{fix}} \in \Sigma$ above is the constant arbitrarily fixed at the end of Section 4.1.)

► **Lemma 17** (preservation of meaning). *If $\Sigma; \vdash t \triangleright (; e)$, then we have $\#_{a_{\text{fix}}}(t) = \llbracket e \theta_\Sigma^{a_{\text{fix}}} \rrbracket$.*

The above lemma follows from a usual substitution lemma (on internal variables) and a subject reduction property; see the full version for the proof.

The correctness of the transformation is stated as the following lemma.

► **Lemma 18** (ordering reflection). *Let: Σ be an alphabet; $a_{\text{fix}} \in \Sigma$; Γ be an environment of the form*

$$\Gamma = \varphi_1 : \kappa_1, \dots, \varphi_m : \kappa_m, f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_\ell : \circ^{q_\ell} \rightarrow \circ$$

where $\mathbf{order}(\kappa_i) = 2$ and $q_i \geq 0$; $t, t' \in \Lambda_{\Gamma, \circ}^\emptyset$; and

$$\Gamma; \vdash t \triangleright (; e) \quad \Gamma; \vdash t' \triangleright (; e').$$

Then we have:

$$t \preceq_{\Gamma, \circ}^{\#, \Sigma, a_{\text{fix}}} t' \quad \text{if} \quad e \preceq_{\Gamma^\sharp, \circ}^{\mathbb{N}} e'.$$

The proof of the above lemma is given in the full version, where we use Lemma 17 and substitution lemmas on external variables.

4.3 $\preceq^{\mathbb{N}}$ on order-2 terms is a wqo

The main goal of this subsection is to prove the following lemma.

► **Lemma 19** ($\preceq_{\Gamma, \circ}^{\mathbb{N}}$ on order-2 terms is wqo). *For $\Gamma = f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_n : \circ^{q_n} \rightarrow \circ$, the quasi-ordering $\preceq_{\Gamma, \circ}^{\mathbb{N}}$ on $\Lambda_{\Gamma, \circ}^{\Sigma_{\mathbb{N}}}$ is a wqo.*

Lemma 15 follows as a corollary of Lemma 19 above and Lemma 18 in the previous subsection:

Proof of Lemma 15. Let $t_0, t_1, \dots \in \Lambda_{\Gamma, \circ}^{\emptyset}$ be an infinite sequence. We have the infinite sequence $e_0, e_1, \dots \in \Lambda_{\Gamma, \circ}^{\Sigma_{\mathbb{N}}}$ such that $\Gamma \vdash t_i \triangleright (; e_i)$, and by Lemma 18, $t_i \preceq_{\Gamma, \circ}^{\#, \Sigma, a_{\text{fix}}} t_j$ if $e_i \preceq_{\Gamma, \circ}^{\mathbb{N}} e_j$. By Lemma 19, there indeed exist i, j ($i < j$) such that $e_i \preceq_{\Gamma, \circ}^{\mathbb{N}} e_j$. Thus, we have $t_i \preceq_{\Gamma, \circ}^{\#, \Sigma, a_{\text{fix}}} t_j$ as required. ◀

To prove Lemma 19, we restrict (without loss of generality) $\Lambda_{\Gamma, \circ}^{\Sigma_{\mathbb{N}}}$ to the set of β -normal forms (which we call *order-2 polynomials*), generated by the following grammar:

$$P ::= 0 \mid 1 \mid P_1 + P_2 \mid P_1 \times P_2 \mid f P_1 \cdots P_q$$

Here, in $f P_1 \cdots P_q$, f should have type $\circ^q \rightarrow \circ$. We write $P_2^{\mathbb{N}}$ for the set of all order-2 polynomials, and write $P_{\Gamma, \circ}^{\mathbb{N}}$ for $\Lambda_{\Gamma, \circ}^{\Sigma_{\mathbb{N}}} \cap P_2^{\mathbb{N}}$. Note that the arity of f may be 0, so that, for example, $f_1(f_2 \times (f_2 + 1)) \in P_{f_1: \circ \rightarrow \circ, f_2: \circ \rightarrow \circ}^{\mathbb{N}}$. Thus, for Lemma 19, the following suffices:

► **Lemma 20** ($\preceq_{\Gamma, \circ}^{\mathbb{N}}$ on order-2 polynomials is wqo). *For $\Gamma = f_1 : \circ^{q_1} \rightarrow \circ, \dots, f_n : \circ^{q_n} \rightarrow \circ$, the quasi-ordering $\preceq_{\Gamma, \circ}^{\mathbb{N}}$ on $P_{\Gamma, \circ}^{\mathbb{N}}$ is a wqo.*

The idea for proving this lemma is as follows:

- An order-2 polynomial is regarded as a tree. Thus, by Kruskal's tree theorem (Proposition 8), the set $P_{\Gamma, \circ}^{\mathbb{N}}$ is well-quasi-ordered with respect to the homeomorphic embedding $\preceq_{\circ}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma}$. Unfortunately, however, the relation $P_1 \preceq_{\circ}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma} P_2$ does not necessarily imply $\preceq_{\Gamma, \circ}^{\mathbb{N}}$; for example, if $P_1 = 1$ and $P_2 = f_1(1)$, then $P_1 \preceq_{\circ}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma} P_2$ holds but $P_1 \not\preceq_{\Gamma, \circ}^{\mathbb{N}} P_2$ does not, because f_1 may be instantiated to $\lambda x.0$. Similarly for $P_1 = f_2$ and $P_2 = f_2 \times 0$.
- To address the problem above, we classify the values of $f \in P_{\Gamma, \circ}^{\mathbb{N}}$ (i.e. elements of $\Lambda_{\circ^q \rightarrow \circ}^{\Sigma_{\mathbb{N}}}$) into a finite number of equivalence classes $A^{(1)}, \dots, A^{(\ell)}$, and use the classification to further normalize order-2 polynomials, so that $P_1 \preceq_{\circ}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma} P_2$ implies $P_1 \preceq_{\Gamma, \circ}^{\mathbb{N}} P_2$ on the normalized polynomials. For example, in the case of $P_1 = 1$ and $P_2 = f_1(1)$ above, the values of f_1 are classified to (i) those that use the argument, (ii) those that return a positive constant without using the argument, and (iii) those that always return 0. We can then normalize $P_2 = f_1(1)$ to $f_1(1)$ (in case (i)), $f_1(0)$ (in case (ii)), and 0 (in case (iii)), respectively. (In case (ii), any argument is replaced with 0, because the argument is irrelevant.) Thus, we can indeed deduce $P_1 \preceq_{\Gamma, \circ}^{\mathbb{N}} P_2$ from $P_1 \preceq_{\circ}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma} P_2$ when the value of f_1 is restricted to just those in (i); and the same holds also for (ii) and (iii). It follows that the restriction of the relation $\preceq_{\Gamma, \circ}^{\mathbb{N}}$ to each classification of the values of $f_1, \dots, f_\ell \in \text{dom}(\Gamma)$ is a wqo. Since the number of classifications is finite, by Dickson's theorem (recall the sentence below Proposition 8), $\preceq_{\Gamma, \circ}^{\mathbb{N}}$ (which is the intersection of the restrictions of $\preceq_{\Gamma, \circ}^{\mathbb{N}}$ to the finite number of classifications) is also a wqo.

We first formalize and justify the reasoning in the last part (using Dickson's theorem).

► **Definition 21** (finite case analysis). For $\Gamma = f_1 : \kappa_1, \dots, f_n : \kappa_n$, we call a *finite case analysis* of Γ a family $(A_i^j)_{i \leq n, j \in J_i}$ of sets such that $\Lambda_{\kappa_i}^{\Sigma_N} = \bigcup_{j \in J_i} A_i^j$ for each $i \leq n$. For $(A_i)_{i \leq n}$ such that $A_i \subseteq \Lambda_{\kappa_i}^{\Sigma_N}$, we define a quasi-ordering $\preceq_{\Gamma, (A_i)_i}^N$ on $\Lambda_{\Gamma, \mathbf{o}}^{\Sigma_N}$ as follows:

$$t \preceq_{\Gamma, (A_i)_i}^N t' \iff \forall t_1 \in A_1, \dots, t_n \in A_n. \llbracket t[t_i/f_i]_i \rrbracket \leq \llbracket t'[t_i/f_i]_i \rrbracket$$

We often omit the subscript Γ of $\preceq_{\Gamma, (A_i)_i}^N$ and write $\preceq_{(A_i)_i}^N$.

The following lemma follows immediately from the fact that the intersection of a finite number of wqo's is a wqo (which is in turn an immediate corollary of Dickson's theorem). (see the full version for omitted proofs in the rest of this section).

► **Lemma 22.** For $\Gamma = f_1 : \kappa_1, \dots, f_n : \kappa_n$ and a finite case analysis $(A_i^j)_{i \leq n, j \in J_i}$ of Γ , if $\preceq_{(A_i^j)_i}^N$ on $\Lambda_{\Gamma, \mathbf{o}}^{\Sigma_N}$ is a wqo for any “case” $(j_i)_{i \leq n} \in \prod_{i \leq n} J_i$, then so is \preceq^N on $\Lambda_{\Gamma, \mathbf{o}}^{\Sigma_N}$.

Thus, to prove Lemma 20, it remains to find an appropriate decomposition $\Lambda_{\kappa_i}^{\Sigma_N} = \bigcup_{j \in J_i} A_i^j$ (where κ_i is an order-1 type $\mathbf{o}^q \rightarrow \mathbf{o}$), and prove that $\preceq_{(A_i^j)_i}^N$ is a wqo.

Henceforth we identify an element of $\Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N}$ with the corresponding element of the polynomial semi-ring $\mathbb{N}[x_1, \dots, x_q]$. For example, $\lambda x_1. \lambda x_2. ((\lambda y. y)x_1) + x_2 \times x_2$ is identified with the polynomial $x_1 + x_2^2$ (which is obtained by normalizing and omitting λ -abstractions, assuming a fixed ordering of the bound variables). For $t \in \Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N}$ we write $\text{poly}(t)$ for the corresponding polynomial.

We define the equivalence relation \sim as the least semi-ring congruence relation on $\mathbb{N}[x_1, \dots, x_q]$ that satisfies (i) $a \sim 1$ if $a > 0$ and (ii) $x_i^j \sim x_i$ if $j > 0$. For example, $2x_1^2x_2 + 3x_1x_2^2 + x_1 + 4 \sim x_1x_2 + x_1 + 1$, and the quotient set $\mathbb{N}[x_1]/\sim$ consists of:

$$[0] \sim, [1] \sim, [x_1] \sim, [x_1 + 1] \sim,$$

and $\mathbb{N}[x_1, x_2]/\sim$ consists of

$$[0] \sim, [1] \sim, [x_1] \sim, [x_2] \sim, [x_1x_2] \sim, [1+x_1] \sim, [1+x_2] \sim, [1+x_1x_2] \sim, [x_1+x_2] \sim, \dots, [1+x_1+x_2+x_1x_2] \sim.$$

In general, $\mathcal{P}(\mathcal{P}([q]))$ (where $[q]$ denotes $\{1, \dots, q\}$ and $\mathcal{P}(X)$ denotes the powerset of X) gives a complete representation of the quotient set $\mathbb{N}[x_1, \dots, x_q]/\sim$, i.e.,

$$\mathbb{N}[x_1, \dots, x_q]/\sim = \left\{ \left[\sum_{\{p_1 < \dots < p_r\} \in \Phi} x_{p_1} \cdots x_{p_r} \right] \sim \mid \Phi \in \mathcal{P}(\mathcal{P}([q])) \right\}.$$

Through $\text{poly} : \Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N} \rightarrow \mathbb{N}[x_1, \dots, x_q]$, we can induce an equivalence relation on $\Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N}$ from \sim on $\mathbb{N}[x_1, \dots, x_q]$, and let A_q^Φ be the equivalence class corresponding to Φ , i.e.,

$$A_q^\Phi := \left\{ t \in \Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N} \mid \text{poly}(t) \sim \sum_{\{p_1 < \dots < p_r\} \in \Phi} x_{p_1} \cdots x_{p_r} \right\}. \quad (1)$$

Then we have $\Lambda_{\mathbf{o}^q \rightarrow \mathbf{o}}^{\Sigma_N} = \bigsqcup_{\Phi \in \mathcal{P}(\mathcal{P}([q]))} A_q^\Phi$. Now, given $\Gamma = f_1 : \mathbf{o}^{q_1} \rightarrow \mathbf{o}, \dots, f_n : \mathbf{o}^{q_n} \rightarrow \mathbf{o}$, we have obtained a finite case analysis of Γ as $(A_{q_i}^\Phi)_{i \leq n, \Phi \in \mathcal{P}(\mathcal{P}([q_i]))}$; for $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$, we write $\preceq_{(\Phi_i)_i}^N$ for $\preceq_{(A_{q_i}^\Phi)_i}^N$. Thus it remains to show that $\preceq_{(\Phi_i)_i}^N$ on $\Lambda_{\Gamma, \mathbf{o}}^{\Sigma_N}$ is a wqo for each $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$.

The following lemma justifies the partition of polynomials based on \sim .

► **Lemma 23** (zero/positive). For any $\Gamma = f_1 : \mathbf{o}^{q_1} \rightarrow \mathbf{o}, \dots, f_n : \mathbf{o}^{q_n} \rightarrow \mathbf{o}$, $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$, and $\Gamma \vdash P : \mathbf{o}$, we have either $P \preceq_{(\Phi_i)_i}^N 0$ or $1 \preceq_{(\Phi_i)_i}^N P$.

In other words, the lemma above says that, given an order-2 polynomial P , whether $P[t_1/f_1, \dots, t_n/f_n]$ evaluates to 0 or not is solely determined by the equivalence classes t_1, \dots, t_n belong to.

► **Example 24.** Let $\Gamma := f : \mathfrak{o}^2 \rightarrow \mathfrak{o}$, and $\Phi := \{\emptyset, \{1, 2\}\} \in \mathcal{P}(\mathcal{P}([2]))$, which denotes the equivalence class $[1 + x_1x_2]_{\sim}$. We have $1 \preceq_{\Phi}^{\mathbb{N}} f P_1 P_2$ for any P_1 and P_2 , since any element of the equivalence class is of the form $a + \dots$ for some natural number $a \geq 1$.

Based on the property above, we define the rewriting relation $\longrightarrow_{(\Phi_i)_i}$, to simplify order-2 polynomials by replacing (i) subterms that always evaluate to 0, and (ii) arguments of a function that are irrelevant, with 0.

► **Definition 25** (rewriting relation and $(\Phi_i)_i$ -normal form). For $\Gamma = f_1 : \mathfrak{o}^{q_1} \rightarrow \mathfrak{o}, \dots, f_n : \mathfrak{o}^{q_n} \rightarrow \mathfrak{o}$ and $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$, we define the relation $\longrightarrow_{(\Phi_i)_i}$ by the following two rules.

- $P \longrightarrow_{(\Phi_i)_i} 0$ if $P \preceq_{(\Phi_i)_i}^{\mathbb{N}} 0$ and $P \neq 0$.
- $f_{\ell} P_1 \dots P_{q_{\ell}} \longrightarrow_{(\Phi_i)_i} f_{\ell} P_1 \dots P_{k-1} 0 P_{k+1} \dots P_{q_{\ell}}$ if (i) $P_k \neq 0$ and (ii) for all $\phi \in \Phi_{\ell}$ such that $k \in \phi$, there exists $p \in \phi$ such that $P_p \preceq_{(\Phi_i)_i}^{\mathbb{N}} 0$.

We write $P_0 \longrightarrow_{(\Phi_i)_i} P_1$ if $P_i = E[P'_i]$ and $P'_0 \longrightarrow_{(\Phi_i)_i} P'_1$ for some E, P'_0 and P'_1 , where the evaluation context E is defined by:

$$E ::= [] \mid E + P \mid P + E \mid E \times P \mid P \times E \mid f P_1 \dots P_{i-1} E P_{i+1} \dots P_q.$$

We call a normal form of $\longrightarrow_{(\Phi_i)_i}$ a $(\Phi_i)_i$ -normal form.

Intuitively, the condition (ii) in the second rule says that whenever the k -th argument P_k is used by f_{ℓ} , it occurs only in the form of $P_k \times P_p \times \dots$ (up to equivalence) and P_p always evaluates to 0; thus, the value of P_k is actually irrelevant.

► **Example 26.** We continue Example 24. Recall $\Gamma = f : \mathfrak{o}^2 \rightarrow \mathfrak{o}$ and $\Phi = \{\emptyset, \{1, 2\}\}$. Consider the order-2 polynomial $f 1 (1 \times 0)$. It can be rewritten to $f 1 0$ by using the first rule (and the evaluation context $E = f 1 []$). We can further apply the second rule to obtain $f 1 0 \longrightarrow_{\Phi} f 0 0$, because $k = 1$ satisfies the conditions ((i) and) (ii). In fact, if $1 \in \phi \in \Phi$, then $\phi = \{1, 2\}$; hence, the required condition holds for $p = 2$. Note that $f 0 0$ is a Φ -normal form; the first rule is not applicable, as $f 0 0 \not\preceq_{\Phi}^{\mathbb{N}} 0$ by the discussion in Example 24.

The following lemma guarantees that any order-2 polynomial can be transformed to at least one equivalent $(\Phi_i)_i$ -normal form.

► **Lemma 27** (existence of normal form).

1. $\longrightarrow_{(\Phi_i)_i}$ is strongly normalizing.
2. If $P \longrightarrow_{(\Phi_i)_i} P'$ then $P \approx_{(\Phi_i)_i}^{\mathbb{N}} P'$.

We can reduce the wqoness of $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$ to that of $\preceq_{\mathfrak{o}}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma}$ by the following lemma:

► **Lemma 28.** For $\Gamma = f_1 : \mathfrak{o}^{q_1} \rightarrow \mathfrak{o}, \dots, f_n : \mathfrak{o}^{q_n} \rightarrow \mathfrak{o}$, $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$, and $(\Phi_i)_i$ -normal forms $\Gamma \vdash P', P : \mathfrak{o}$, if $P' \preceq_{\mathfrak{o}}^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma} P$ then $P' \preceq_{(\Phi_i)_i}^{\mathbb{N}} P$.

The proof is given by a simple calculation using Lemma 23 and that the given $(\Phi_i)_i$ -normal forms P', P do not satisfy the condition for the rewriting $\longrightarrow_{(\Phi_i)_i}$.

Now we are ready to prove Lemma 20.

Proof of Lemma 20. By Lemma 22, it suffices to show that $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$ on $P_{\Gamma,o}^{\mathbb{N}}$ is a wqo for each $(\Phi_i)_i \in \prod_{i \leq n} \mathcal{P}(\mathcal{P}([q_i]))$. By the Kruskal’s tree theorem, $\preceq_o^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma}$ on $P_{\Gamma,o}^{\mathbb{N}}$ is a wqo, and hence the sub-ordering $\preceq_o^{\text{he}, \Sigma_{\mathbb{N}} \cup \Gamma}$ on the subset

$$\{P \in P_{\Gamma,o}^{\mathbb{N}} \mid P \text{ is a } (\Phi_i)_i\text{-normal form}\} \subseteq P_{\Gamma,o}^{\mathbb{N}}$$

is a wqo. Therefore by Lemma 28, $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$ on $\{P \in P_{\Gamma,o}^{\mathbb{N}} \mid P \text{ is a } (\Phi_i)_i\text{-normal form}\}$ is a wqo. By Lemma 27, $\{P \in P_{\Gamma,o}^{\mathbb{N}} \mid P \text{ is a } (\Phi_i)_i\text{-normal form}\}$ and $P_{\Gamma,o}^{\mathbb{N}}$ – both modulo $\beta\eta$ -equivalence – are isomorphic (with respect to $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$ and $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$); hence $\preceq_{(\Phi_i)_i}^{\mathbb{N}}$ on $P_{\Gamma,o}^{\mathbb{N}}$ is a wqo. ◀

5 Conclusion

We have introduced the nAK-conjecture, a weaker version of the AK-conjecture in [2], and proved it up to order 3. We have also proved a pumping lemma for higher-order grammars (which is slightly weaker than the pumping lemma conjectured in [2]) under the assumption that the nAK-conjecture holds. Obvious future work is to show the nAK-conjecture or the original AK-conjecture for arbitrary orders. Finding other applications of the two conjectures (cf. an application of Kruskal’s tree theorem to program termination [4]) is also left for future work.

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 Programming (ICALP 2016)*, volume 55 of *LIPIcs*, pages 111:1–111:13. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2016.
- 2 Kazuyuki Asada and Naoki Kobayashi. Pumping Lemma for Higher-order Languages. In Ioannis Chatzigiannakis, Piotr Indyk, Fabian Kuhn, and Anca Muscholl, editors, *44th International Colloquium on Automata, Languages, and Programming, ICALP 2017, July 10–14, 2017, Warsaw, Poland*, volume 80 of *LIPIcs*, pages 97:1–97:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.ICALP.2017.97.
- 3 Werner Damm. The IO- and OI-Hierarchies. *Theor. Comput. Sci.*, 20:95–207, 1982.
- 4 Nachum Dershowitz. Orderings for term-rewriting systems. *Theoretical Computer Science*, 17(3):279–301, 1982. doi:10.1016/0304-3975(82)90026-3.
- 5 Takeshi Hayashi. On Derivation Trees of Indexed Grammars –An Extension of the uvwxy-Theorem–. *Publ. RIMS, Kyoto Univ.*, pages 61–92, 1973.
- 6 Graham Higman. Ordering by divisibility in abstract algebras. *Proceedings of the London Mathematical Society*, 3(1):326–336, 1952.
- 7 Joseph 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>.
- 8 Pawel Parys. Intersection Types and Counting. In Naoki Kobayashi, editor, *Proceedings Eighth Workshop on Intersection Types and Related Systems, ITRS 2016, Porto, Portugal, 26th June 2016.*, volume 242 of *EPTCS*, pages 48–63, 2016. doi:10.4204/EPTCS.242.6.
- 9 Pawel Parys. The Complexity of the Diagonal Problem for Recursion Schemes. In Satya V. Lokam and R. Ramanujam, editors, *37th IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2017, December 11–15, 2017, Kanpur, India*, volume 93 of *LIPIcs*, pages 45:1–45:14. Schloss Dagstuhl – Leibniz-Zentrum fuer Informatik, 2017. doi:10.4230/LIPIcs.FSTTCS.2017.45.

- 10 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.
- 11 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.
- 12 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.
- 13 Marek Zaionc. Lambda Representation of Operations Between Different Term Algebras. In Leszek Pacholski and Jerzy Tiuryn, editors, *Computer Science Logic, 8th International Workshop, CSL '94, Kazimierz, Poland, September 25-30, 1994, Selected Papers*, volume 933 of *Lecture Notes in Computer Science*, pages 91–105. Springer, 1994. doi:10.1007/BFb0022249.