ISSN 1346-5597

NII Technical Report

Distributional Learning of Some Nonlinear Tree Grammars

Alexander Clark, Makoto Kanazawa, Gregory M. Kobele, and Ryo Yoshinaka

NII-2015-004E
Oct. 2015

Distributional Learning of Some Nonlinear Tree Grammars

Alexander Clark[*]
Department of Philosophy
King's College London

Makoto Kanazawa[*] [C]
National Institute of Informatics, Tokyo, and
SOKENDAI (Graduate University for Advanced Studies)

Gregory M. Kobele[*]
Department of Linguistics and Computation Institute
University of Chicago

Ryo Yoshinaka[*]
Graduate School of Informatics
Kyoto University

Abstract. A key component of Clark and Yoshinaka's distributional learning algorithms is the extraction of substructures and contexts contained in the input data. This problem often becomes intractable with nonlinear grammar formalisms due to the fact that more than polynomially many substructures and/or contexts may be contained in each object. Previous works on distributional learning of nonlinear grammars avoided this difficulty by restricting the substructures or contexts that are made available to the learner. In this paper, we identify two classes of nonlinear tree grammars for which the extraction of substructures and contexts can be performed in polynomial time, and which, consequently, admit successful distributional learning in its unmodified, original form.

Keywords: Distributional learning, tree language, tree pattern, generalized context-free grammar, parallel regular tree grammar, IO context-free tree grammar

1. Introduction

In *distributional learning* [1, 13, 12, 6, 15, 2], we are concerned with learning sets of discrete combinatorial objects of various types: strings, trees, λ-terms, graphs, etc. An important step in all these algorithms is *decomposition*: we decompose an object into two parts—some piece or *substructure* of that object and the surrounding *context* in which that piece occurs in the whole object. The precise form

[*]This work was supported by NII joint research project "Algorithmic Learning of Nonlinear Formalisms Based on Distributional Learning".
[C]Corresponding author

of substructures and contexts is determined by the nature of the grammars hypothesized by the learner. In broadly "context-free" grammar formalisms, we can distinguish between the derivations, which are tree-structured, and the objects which they derive, which may not be. In this case, the substructures of interest are those which correspond to subtrees of a possible derivation tree for that object, and a context corresponds to the contribution of the remainder of the derivation tree. For example, in the case of ordinary context-free grammars generating strings, a substructure of a string is any substring of that string, and the corresponding context is represented by the prefix-suffix pair that surrounds that substring. In distributional learning of other kinds of "context-free" grammars, we might decompose a tree into a subtree and a tree context (i.e., tree with a "hole"), or a λ-term of type β into a λ-term of type α and a λ-term of type $\alpha \to \beta$.

The substructures and contexts are used in different ways in the two approaches to distributional learning, known as the *primal* and *dual* approaches. In the primal approach, each nonterminal in the hypothesized grammar is indexed by a finite set consisting of a bounded number of substructures extracted from the input positive data. These nonterminals, together with "grammatical operations" allowed by the grammar class (e.g., concatenation of strings), are used to build candidate rules. The contexts extracted from the input data are then used to filter out those rules that are incompatible with the requirement that the distribution of the substructures that are derived from each nonterminal A be *characterized* by the distribution of the finite set of substructures associated with A. Take, for example, the simple case of ordinary context-free grammars. When nonterminals A_0, A_1, A_2 in the hypothesized grammar are indexed by finite sets $\mathbf{S}_0, \mathbf{S}_1, \mathbf{S}_1$ of strings, the rule

$$A_0 \to A_1 A_2$$

is deemed incorrect if there is a prefix-suffix pair (u, v) among the contexts extracted from the input data such that $u\mathbf{S}_0v$ is included in the target language but $u\mathbf{S}_1\mathbf{S}_2v$ is not. (This is determined by querying the *membership oracle* for the target language, which is assumed to be available to the learner.)

The dual approach to distributional learning swaps the roles of the substructures and the contexts, with the nonterminals being indexed by finite sets (of bounded cardinality) of extracted contexts and the extracted substructures used to eliminate the bad rules. For example, if nonterminals A_0, A_1, A_2 of the hypothesized context-free grammar are indexed by finite sets $\mathbf{C}_0, \mathbf{C}_1, \mathbf{C}_2$ of prefix-suffix pairs, the rule $A_0 \to A_1 A_2$ is rejected when there are substrings s_1, s_2 among those extracted from the input data such that every context in $\mathbf{C}_1$ can appear flanking s_1 in the target language and likewise for $\mathbf{C}_2$ and s_2, and yet there is a pair (u, v) in $\mathbf{C}_0$ such that us_1s_2v is not in the target language.

To sum up, distributional learning in its pristine form uses the sets $\mathbb{S}|_D$ and $\mathbb{C}|_D$ of substructures and contexts extracted from input data D: one to characterize nonterminals, the other to filter out hypothesized rules. The efficient computation of these two sets—the problem of decomposition—is crucial in each of the two approaches to distributional learning.

In some cases, decomposition is algorithmically trivial: for example, there are only $\binom{n+1}{2} + n + 1 = \frac{(n+2)(n+1)}{2}$ different ways to divide a string of length n into a substring and a prefix-suffix pair, and enumerating all of them is simply a matter of picking all unordered pairs of inter-symbol positions inside a given string. More generally, decomposition is tractable when the available grammatical operations are *linear*. The linearity of grammatical operations means that the combination of a substructure and a context never deletes or duplicates any part of the substructure or the context. In this case, the number of ways to divide a given object into a possible substructure and a possible context is (when certain parameters are fixed) bounded by a polynomial in the size of the given object, and all these different ways of decomposition can be enumerated in polynomial time [15]. In contrast, when grammatical operations are allowed to be nonlinear—in other words, when the combination of a substructure with a context may duplicate some part of the substructure or the context—the number of different decompositions will in general not be polynomially bounded; in this case it will be impossible to explicitly enumerate them all in polynomial time.

This phenomenon of non-polynomiality of decomposition already manifests itself with a very simple class of *parallel multiple context-free grammars*: the class $\mathbb{G}(1,1,2)$ in Clark and Yoshinaka's [2] symbolism. Rules of grammars in this class have one of the following forms, where A, B are nonterminals, $w \in \Sigma^*$, $x \notin \Sigma$ is a variable, and $\pi \in \Sigma^* x \Sigma^* \cup \Sigma^* x \Sigma^* x \Sigma^*$:

$$\begin{aligned} A(w) &:- \\ A(\pi) &:- B(x) \end{aligned}$$

Contexts in this case are certain elements of $(\Sigma^* x)^+ \Sigma^*$ (representing functions from Σ^* to Σ^*), substructures are just strings in Σ^*, and the combination of a context c and a substructure s consists in substituting the string s for the variable x in c. For example, when $a \in \Sigma$, the following are some of the possible rules ($k, l \geq 0$):

$$\begin{aligned} A(xa^k x) &:- A(x) \\ A(xa^l) &:- A(x) \end{aligned}$$

Let us denote a rule of the first form by ρ_k and one of the second form by σ_l. Chaining $\rho_{k_m}, \dots, \rho_{k_1}$ in this order forms an incomplete derivation that contributes a context $c_{(k_1,\dots,k_m)}$ defined by

$$\begin{aligned} c_{()} &= x, \\ c_{(k_1,\dots,k_{m-1},k_m)} &= c_{(k_1,\dots,k_{m-1})} a^{k_m} c_{(k_1,\dots,k_{m-1})} \quad \text{for } m \geq 1. \end{aligned}$$

When $k_i \leq 2^m - 1$ for each $i = 1, \dots, m$, the length of $c_{(k_1,\dots,k_m)}$ is

$$\sum_{i=1}^{m} 2^{m-i} k_i + 2^m \leq \sum_{i=1}^{m} 2^{m-i}(2^m - 1) + 2^m < 2^{2m},$$

and with an appropriate value of l, the sequence of rules $\sigma_l, \rho_{k_m}, \dots, \rho_{k_1}$ always gives a context $c_{(k_1,\dots,k_m)} a^l$ which can combine with a substructure a to form the string $a^{2^{2m}}$. But all these contexts $c_{(k_1,\dots,k_m)} a^l$ are different for different choices of $k_1, \dots, k_m$, which means that there are $(2^m)^m = 2^{m^2}$ of them. It follows that the number of these contexts is not bounded by any polynomial in the length $n = 2^{2m}$ of the string $a^{2^{2m}}$.

Despite this difficulty, Clark and Yoshinaka [2] succeeded in giving a kind of distributional learning algorithm for parallel multiple context-free grammars. How did they overcome the non-polynomiality of decomposition? As we have seen, when the learning target is the class $\mathbb{G}(p, q, r)$ of parallel multiple context-free grammars with dimension bound $p \geq 1$, rank bound $q \geq 1$, and copying degree bound $r \geq 2$, the set $\mathbb{C}|_D$ of contexts contained in the input data D cannot be enumerated in polynomial time. However, the set $\mathbb{S}|_D$ of substructures contained in D simply consists of tuples of strings with at most p components and is quite easy to enumerate. Clark and Yoshinaka's solution was to take the dual approach, using only a small subset of $\mathbb{C}|_D$ to form characterizing sets of contexts for nonterminals.

Thus, they demanded not only that each nonterminal of the target grammar be characterized by a finite set of contexts of bounded cardinality, but also that each element of the characterizing set be an r-copying context (i.e., string pattern where each variable occurs at most r times). When a PMCFG in $\mathbb{G}(p, q, r)$ has a finite characterizing set of contexts for each of its nonterminals, there is no reason to expect the contexts involved to be r-copying, so this algorithm misses many grammars in $\mathbb{G}(p, q, r)$ that have a bounded-cardinality characterizing set of contexts for each of their nonterminals.[1] Crucially, the algorithm needs $\mathbb{S}|_D$ in its entirety to correctly filter out bad rules. For this reason, it does not seem to be possible to take the primal approach to learning PMCFGs based on a similar idea.[2]

In this paper, we show, for the first time, the existence of non-trivial classes of nonlinear grammars for which the decomposition problem is solvable in polynomial time, and which, consequently, admit both primal and dual distributional learning algorithms in their pristine form. We present two: the classes of *parallel regular tree grammars* and of *uniformly copying IO context-free tree grammars*. The proof of the polynomial complexity of decomposition is far from trivial, and in both cases relies on a combinatorial lemma about certain kinds of tree patterns that match a given tree.

Rather than establishing the learnability results for the two cases separately, we first present distributional learning algorithms in an abstract form and give a set of general conditions that are jointly sufficient for these algorithms to succeed on any broadly context-free grammar formalism. The proof of learnability for the particular cases will then reduce to checking that these conditions are satisfied.

2. Distributional Learning in a General Setting

We first consider an abstract form of distributional learning on broadly context-free grammar formalisms and establish general conditions that are sufficient for successful learning. In dealing with concrete grammar classes in later sections, all we need to do is to check that these conditions are satisfied.

2.1. Generalized Context-Free Grammars

The following definition slightly deviates from [9, 10]:

Definition 2.1. A *generalized context-free grammar* [9, 10] is a 6-tuple $G = (N, O, F, \sigma, P, I)$ where

1. N is a finite set of *nonterminals*,
2. O is a countable set of *objects*,
3. $F = \bigcup_{q \in \mathbb{N}} F_q$ is a finite set of partial functions, where F_q is a finite set of partial functions from O^q to O,
4. σ is a function from N to $\mathscr{P}(O)$,

[1]Also, "r-copying" is an arbitrary restriction; other values, e.g. 2^r, would work just as well.

[2]A recent paper by Kanazawa and Yoshinaka [5] pushes this strategy further in the general setting of tree-generating *almost linear second-order abstract categorial grammars* with a bound on the degree of nonlinearity either in the substructures or in the contexts.

5. P is a finite subset of $\bigcup_{q \in \mathbb{N}}(F_q \times N^{q+1})$ such that for each $(f, A_0, A_1, \ldots, A_q) \in P$,
$$\sigma(A_1) \times \cdots \times \sigma(A_q) \subseteq \mathrm{dom}(f),$$
$$f(\sigma(A_1) \times \cdots \times \sigma(A_q)) \subseteq \sigma(A_0),$$
and

6. I is a subset of N.

We adopt the convention that when $q = 0$, a partial function from O^q to O is simply an element of O. The value $\sigma(A)$ of σ on $A \in N$ is called the *sort* of A. The elements of I are called *initial nonterminals*, and the elements of P are called *rules*.

A rule $(f, A_0, A_1, \ldots, A_q)$ is written sometimes in the form of a rewriting rule
$$A_0 \to f[A_1, \ldots, A_q]$$
and sometimes as a Horn clause
$$A_0(f(z_1, \ldots, z_q)) :- A_1(z_1), \ldots, A_q(z_q)$$
where in place of $f(z_1, \ldots, z_q)$, we may write an expression defining the value of $f(z_1, \ldots, z_q)$. We use the latter notation.[3]

A rule $A_0(f(z_1, \ldots, z_q)) :- A_1(z_1), \ldots, A_q(z_q)$ is called a *terminating rule* when $q = 0$.

The notion of an *A-derivation tree* and its *yield* are defined inductively as follows:

- If $\pi = A_0(f) :-$ is a terminating rule, then π is an A_0-derivation tree, and its yield is $f \in O$.
- If $\pi = A_0(f(z_1, \ldots, z_q)) :- A_1(z_1), \ldots, A_q(z_q)$ is a non-terminating rule ($q \geq 1$) and for each $i = 1, \ldots, q$, τ_i is an A_i-derivation tree with yield S_i, then $\pi(\tau_1, \ldots, \tau_q)$ is an A_0-derivation tree and its yield is $f(S_1, \ldots, S_q)$.

Note that the yield of an A-derivation tree is always defined and belongs to $\sigma(A)$. We call such an object *A-substructure*, and write $\mathcal{S}(G, A)$ for the set of A-substructures. When $A \in I$, an A-derivation tree is called a *complete derivation tree*. The *language* of G, written $\mathcal{L}(G)$, is $\bigcup_{A \in I} \mathcal{S}(G, A)$, or the set of yields of complete derivation trees of G.

For each nonterminal $A \in N$, we introduce a new symbol $\Box^A$. We call such symbols *holes*. For a finite sequence $(A_1, \ldots, A_n)$ of nonterminals, an *$(A_1, \ldots, A_n)$-derivation environment* and its *yield* are defined inductively as follows:

- If $A \in I$, then $\Box^A$ is an (A)-derivation environment and its yield is the identity function on $\sigma(A)$.
- If ξ is a $(B_1, \ldots, B_i, A_0, B_{i+1}, \ldots, B_n)$-derivation environment with yield C_0 and $\pi = A_0(f(z_1, \ldots, z_q)) :- A_1(z_1), \ldots, A_q(z_q)$ is a rule, then the result of substituting $\pi(\Box^{A_1}, \ldots, \Box^{A_q})$ for the $(i+1)$th hole (which is $\Box^{A_0}$) in ξ is a $(B_1, \ldots, B_i, A_1, \ldots, A_q, B_{i+1}, \ldots, B_n)$-derivation environment and its yield is the function $C \colon \sigma(B_1) \times \cdots \times \sigma(B_i) \times \sigma(A_1) \times \cdots \times \sigma(A_q) \times \sigma(B_{i+1}) \times \cdots \times \sigma(B_n) \to O$ such that $C(z_1, \ldots, z_i, x_1, \ldots, x_q, z_{i+1}, \ldots, z_n) = C_0(z_1, \ldots, z_i, f(x_1, \ldots, x_q), z_{i+1}, \ldots, z_n)$ for every $z_i \in \sigma(B_i)$ ($i = 1, \ldots, n$) and $x_j \in \sigma(A_j)$ ($j = 1, \ldots, q$).

[3] In the rewriting notation, the "f" on the right-hand side is a function symbol which should not be expanded by its definition until the whole derivation is complete.

We call the yield of an (A)-derivation environment an *A-context*, and write $\mathcal{C}(G, A)$ for the set of A-contexts of G. An A-context is always a function from $\sigma(A)$ to O.

If ξ is an $(A_1, \dots, A_n)$-derivation environment with yield C and τ_i is an A_i-derivation tree with yield S_i for $i = 1, \dots, n$, then the result of substituting τ_i for the ith hole (i.e., $\square^{A_i}$) in ξ $(i = 1, \dots, n)$ is a complete derivation tree whose yield is $C(S_1, \dots, S_n)$. In particular, a $()$-derivation environment is nothing but a complete derivation tree.

The set of (A)-derivation environments together with their yield has the following alternative inductive definition:

- If $A \in I$, then $\square^A$ is an (A)-derivation environment and its yield is the identity function on $\sigma(A)$.
- If ξ is an (A_0)-derivation environment with yield C_0, $\pi = A_0(f(z_1, \dots, z_q)) \mathrel{:-} A_1(z_1), \dots, A_q(z_q)$ is a non-terminating rule $(q \geq 1)$, and τ_j is an A_j-derivation tree with yield S_j for $j = 1, \dots, i-1, i+1, \dots, q$ $(1 \leq i \leq q)$, then the result of substituting $\pi(\tau_1, \dots, \tau_{i-1}, \square^{A_i}, \tau_{i+1}, \dots, \tau_q)$ for $\square^{A_0}$ in ξ is an (A_i)-derivation environment, and its yield is the function $C \colon \sigma(A_i) \to O$ such that $C(z) = C_0(f(S_1, \dots, S_{i-1}, z, S_{i+1}, \dots, S_q))$ for every $z \in \sigma(A_i)$.

2.2. Distributional Learning

Our learning paradigm is *identification in the limit from positive data and membership queries*. A *positive presentation* of a language L_* is an infinite sequence $T_1, T_2, \dots$ such that $L_* = \{\, T_i \mid i \geq 1 \,\}$. A learner is given a positive presentation of the language $L_* = \mathcal{L}(G_*)$ of the target grammar G_* and each time a new example T_i is given, it outputs a grammar G_i computed from $T_1, \dots, T_i$ with the aid of a *membership oracle*. One may query the oracle whether a certain object T is in L_*, and the oracle answers in constant time. For a learning algorithm to be successful in learning G_*, it must be that for any positive presentation $T_1, T_2, \dots$ of L_*, there is an integer n such that $G_n = G_m$ for all $m \geq n$ and $\mathcal{L}(G_n) = L_*$. Because of the access to the membership oracle, this condition alone is not sufficiently restrictive; successful distributional learning algorithms must satisfy further favorable properties in terms of efficiency.

2.2.1. Basic Properties and Assumptions

Let us fix a countable set $\mathbb{O}$ of objects with a distinguished subset $\mathbb{O}_0 \subseteq \mathbb{O}$, a countable set $\mathbb{R} \subseteq \mathscr{P}(\mathbb{O})$ of subsets of $\mathbb{O}$, and for each $q \in \mathbb{N}$, a countable set $\mathbb{F}_q$ of partial functions from O^q to O. Let $\mathbb{F} = \bigcup_{q \in \mathbb{N}} \mathbb{F}_q$ and let $\mathbb{G}$ be the class of generalized context-free grammars $G = (N, \mathbb{O}, F, \sigma, P, I)$ such that

- $N \subseteq \mathbb{N}$,
- $F \subseteq \mathbb{F}$,
- $\mathrm{ran}(\sigma) \subseteq \mathbb{R}$, and
- $\sigma(A) \subseteq \mathbb{O}_0$ for all $A \in I$.

(Nonterminals of grammars in $\mathbb{G}$ are formally natural numbers, but in informal description of grammars, we use other finitely presented objects as nonterminals, assuming some suitable Gödel numbering of the nonterminals.)

Relative to the class $\mathbb{G}$, we define the set $\mathbb{S}$ of *substructures* and the set $\mathbb{C}$ of *contexts* by

$$\mathbb{S} = \bigcup \{\, \mathcal{S}(G, A) \mid A \text{ is a nonterminal of some } G \in \mathbb{G} \,\},$$
$$\mathbb{C} = \bigcup \{\, \mathcal{C}(G, A) \mid A \text{ is a nonterminal of some } G \in \mathbb{G} \,\}.$$

Lemma 2.2. Suppose $f \in \mathbb{F}_q, R_0, R_1, \ldots, R_q \in \mathbb{R}$, and

$$R_1 \times \cdots \times R_q \subseteq \operatorname{dom}(f),$$
$$f(R_1 \times \cdots \times R_q) \subseteq R_0.$$

(i) If $S_i \in \mathbb{S} \cap R_i$ for $i = 1, \ldots, q$, then $f(S_1, \ldots, S_q) \in \mathbb{S}$. As a special case, if $q = 0$, then $f \in \mathbb{S}$.

(ii) If $R_0 \subseteq \mathbb{O}_0$, then the identity function on R_0 is in $\mathbb{C}$.

(iii) Suppose $C \in \mathbb{C}$, $R_0 \subseteq \operatorname{dom}(C)$, and $S_i \in \mathbb{S} \cap R_i$ for $i = 1, \ldots, j-1, j+1, \ldots, q$. Define a function $C' \colon R_j \to \mathbb{O}_0$ by

$$C'(z) = C(f(S_1, \ldots, S_{j-1}, z, S_{j+1}, \ldots, S_q)) \quad \text{for every } z \in R_j.$$

Then $C' \in \mathbb{C}$.

Definition 2.3. Let $G = (N, \mathbb{O}, F, \sigma, P, I) \in \mathbb{G}$ and $A \in N$.

(i) We say that a finite nonempty set $\mathbf{S} \subseteq \mathbb{S} \cap \sigma(A)$ is a *kernel* for A if the following equivalence holds for all $C \in \mathbb{C}$ such that $\sigma(A) \subseteq \operatorname{dom}(C)$:[4]

$$\forall S \in \mathcal{S}(G, A)(C(S) \in \mathcal{L}(G)) \Leftrightarrow \bigwedge_{S \in \mathbf{S}} C(S) \in \mathcal{L}(G).$$

When $|\mathbf{S}| \leq m$, we call $\mathbf{S}$ an *m-kernel* for A.

(ii) We say that a finite nonempty set $\mathbf{C} \subseteq \{\, C \in \mathbb{C} \mid \sigma(A) \subseteq \operatorname{dom}(C) \,\}$ is a *context set* for A if the following equivalence holds for all $S \in \mathbb{S} \cap \sigma(A)$:

$$S \in \mathcal{S}(G, A) \Leftrightarrow \bigwedge_{C \in \mathbf{C}} C(S) \in \mathcal{L}(G).$$

When $|\mathbf{C}| \leq m$, we call $\mathbf{C}$ an *m-context set* for A.[5]

[4]We use the symbol $\bigwedge$ to express finite conjunctions. If $\mathbf{S} = \{S_1, \ldots, S_m\}$, then $\bigwedge_{S \in \mathbf{S}} C(S) \in \mathcal{L}(G)$ means $C(S_1) \in \mathcal{L}(G) \wedge \cdots \wedge C(S_m) \in \mathcal{L}(G)$.

[5]Yoshinaka [13] and Leiß [7] used a weaker definition of a context set for ordinary context-free grammars, which can be expressed in the present general setting as follows: for all $S \in \mathbb{S} \cap \sigma(A)$,

$$\forall C' \in \mathbb{C}((\sigma(A) \subseteq \operatorname{dom}(C') \wedge \forall S' \in \mathcal{S}(G, A)(C'(S') \in \mathcal{L}(G))) \Rightarrow C'(S) \in \mathcal{L}(G)) \Leftrightarrow \bigwedge_{C \in \mathbf{C}} C(S) \in \mathcal{L}(G).$$

This weaker definition also verifies Lemma 2.5 below and hence suffices for our dual distributional learning algorithm (Algorithm 2) to work properly.

(iii) We say that G has the *m-finite kernel property* (m-FKP) if every nonterminal of G has an m-kernel.

(iv) We say that G has the *m-finite context property* (m-FCP) if every nonterminal of G has an m-context set.

A *primal learner* for $\mathbb{G}$ tries to learn every grammar in $\mathbb{G}$ with the m-FKP for a certain fixed value of m. In contrast, a *dual learner* for $\mathbb{G}$ tries to learn every grammar in $\mathbb{G}$ with the m-FCP.

Let $T \in \mathbb{O}_0$. We say that $S \in \mathbb{S}$ is *contained in* T if there exists $C \in \mathbb{C}$ such that $C(S)$ is defined and equals T. Likewise, $C \in \mathbb{C}$ is said to be *contained in* T if there exists $S \in \mathbb{S}$ such that $C(S)$ is defined and equals T. For $f \in \mathbb{F}_q$, we say that f is *contained in* $T \in \mathbb{O}_0$ if there are $C \in \mathbb{C}$ and $S_1, \dots, S_q \in \mathbb{S}$ such that $C(f(S_1, \dots, S_q))$ is defined and equals T. For $T \in \mathbb{O}_0$, define

$$\begin{aligned}\mathbb{S}|_T &= \{\, S \in \mathbb{S} \mid S \text{ is contained in } T \,\},\\ \mathbb{C}|_T &= \{\, C \in \mathbb{C} \mid C \text{ is contained in } T \,\},\\ \mathbb{F}|_T &= \{\, f \in \mathbb{F} \mid f \text{ is contained in } T \,\}.\end{aligned}$$

Lemma 2.4. Let $G = (N, \mathbb{O}, F, \sigma, P, I) \in \mathbb{G}$ and $L = \mathcal{L}(G)$. Suppose that each nonterminal $A \in N$ has a kernel $\mathbf{S}_A$.

(i) If $\mathcal{C}(G, A) \neq \varnothing$, then every $S \in \mathbf{S}_A$ is contained in some $T \in L$.

(ii) If $A \in I$, then $\mathbf{S}_A \subseteq L$.

(iii) For every rule $A_0(f(z_1, \dots, z_q))$:– $A_1(z_1), \dots, A_q(z_q)$ in P, the following condition holds:

$$\forall C \in \mathbb{C} \left(\left(\sigma(A_0) \subseteq \mathrm{dom}(C) \wedge \bigwedge_{S_0 \in \mathbf{S}_{A_0}} C(S_0) \in L \right) \Rightarrow \bigwedge_{S_1 \in \mathbf{S}_{A_1}} \cdots \bigwedge_{S_q \in \mathbf{S}_{A_q}} C(f(S_1, \dots, S_q)) \in L \right).$$

Proof:
We only show (iii). Let $C \in \mathbb{C}$ and assume

$$\sigma(A_0) \subseteq \mathrm{dom}(C) \wedge \bigwedge_{S_0 \in \mathbf{S}_{A_0}} C(S_0) \in L.$$

Let $S_1 \in \mathbf{S}_{A_1}, \dots, S_q \in \mathbf{S}_{A_q}$. We show the following claim by induction on $i \in \{0, \dots, q\}$:

$$\forall S'_{i+1} \in \mathcal{S}(G, A_{i+1}) \dots \forall S'_q \in \mathcal{S}(G, A_q)(C(f(S_1, \dots, S_i, S'_{i+1}, \dots, S'_q)) \in L). \tag{1}$$

The case of $i = 0$ is clear since $A_0(f(z_1, \dots, z_q))$:– $A_1(z_1), \dots, A_q(z_q)$ is a rule of G. Now assume that (1) holds for $i = j - 1 < q$, and let $S'_{j+1} \in \mathcal{S}(G, A_{j+1}), \dots, S'_q \in \mathcal{S}(G, A_q)$. Let $C' \colon \sigma(A_j) \to \mathbb{O}_0$ be the function defined by

$$C'(x) = C(f(S_1, \dots, S_{j-1}, x, S'_{j+1}, \dots, S'_q)) \quad \text{for all } x \in \sigma(A_j).$$

By part (iii) of Lemma 2.2, $C' \in \mathbb{C}$. Since (1) holds for $i = j - 1$, we have $C'(S'_j) \in L$ for all $S'_j \in \mathcal{S}(G, A_j)$. Since $S_j \in \mathbf{S}_{A_j}$, it follows that $C'(S_j) \in L$, i.e.,

$$C(f(S_1, \dots, S_j, S'_{j+1}, \dots, S'_q)) \in L.$$

Since $S'_{j+1}, \dots, S'_q$ were arbitrary, it follows that (1) holds for $i = j$. □

Lemma 2.5. Let $G = (N, \mathbb{O}, F, \sigma, P, I) \in \mathbb{G}$ and $L = \mathcal{L}(G)$. Suppose that each nonterminal $A \in N$ has a context set $\mathbf{C}_A$.

(i) If $\mathcal{S}(G, A) \neq \varnothing$, then every $C \in \mathbf{C}_A$ is contained in some $T \in L$.

(ii) If $A \in I$, then the following condition holds:

$$\forall S \in \mathbb{S} \left(\left(S \in \sigma(A) \wedge \bigwedge_{C \in \mathbf{C}_A} C(S) \in L \right) \Rightarrow S \in L \right).$$

(iii) For every rule $A_0(f(z_1, \dots, z_q)) :- A_1(z_1), \dots, A_q(z_q)$, in P, the following condition holds:

$$\forall S_1 \dots S_q \in \mathbb{S} \left(\bigwedge_{i=1}^{q} \left(S_i \in \sigma(A_i) \wedge \bigwedge_{C_i \in \mathbf{C}_{A_i}} C_i(S_i) \in L \right) \Rightarrow \bigwedge_{C_0 \in \mathbf{C}_{A_0}} C_0(f(S_1, \dots, S_q)) \in L \right).$$

We make a number of assumptions that are necessary to make a distributional learning algorithm possible. In the following, we assume that elements of $\mathbb{O}, \mathbb{R}, \mathbb{F}, \mathbb{G}, \mathbb{S}, \mathbb{C}$ are all finitely represented under some effective encoding.

Assumption 1. (Finiteness of the set of sorts)
$\mathbb{R}$ is a finite set of subsets of $\mathbb{O}$.

Assumption 2. (Bound on the arity of functions)
$\mathbb{F}_q$ is empty for all but finitely many q.

Assumption 3. (Polynomial complexity of questions about functions)
Let $R, R_1, \dots, R_q \in \mathbb{R}$.

- For $f \in \mathbb{F}_q$, the question "$R_1 \times \dots \times R_q \subseteq \mathrm{dom}(f)$?" is decidable in polynomial time in the size of f.
- For $f \in \mathbb{F}_q$, the question "$f(R_1 \times \dots \times R_q) \subseteq R$?" is decidable in polynomial time in the size of f;
- For $f \in \mathbb{F}_q$ and $(S_1, \dots, S_q) \in \mathrm{dom}(f)$, $f(S_1, \dots, S_q)$ is computable in polynomial time in the combined size of $f, S_1, \dots, S_q$.

Assumption 4. (Polynomial complexity of universal recognition)
For $T \in \mathbb{O}_0$ and $G \in \mathbb{G}$, the question "$T \in \mathcal{L}(G)$?" is decidable in polynomial time in the combined size of T and G.

Assumption 5. (Polynomial complexity of questions about contexts and substructures)
Let $R \in \mathbb{R}$.

- For $S \in \mathbb{S}$, the question "$S \in R$?" is decidable in polynomial time in the size of S.
- For $C \in \mathbb{C}$, the question "$R \subseteq \mathrm{dom}(C)$?" is decidable in polynomial time in the size of C.
- For $S \in \mathbb{S} \cap R$ and $C \in \mathbb{C}$ such that $R \subseteq \mathrm{dom}(C)$, $C(S)$ is computable in polynomial time in the combined size of C, S.

Assumption 6. (Polynomial enumerability of substructures, contexts, and functions)
Each of $\mathbb{S}|_T, \mathbb{C}|_T, \mathbb{F}|_T$ is finite and can be enumerated in polynomial time in the size of $T \in \mathbb{O}_0$.

We can state the general forms of both primal and dual distributional learning algorithms for any class $\mathbb{G}$ satisfying Assumptions 1, 2, 3, 4, 5 and 6.[6] When D is a finite subset of $\mathbb{O}_0$, we write

$$\mathbb{S}|_D = \bigcup_{T \in D} \mathbb{S}|_T, \quad \mathbb{C}|_D = \bigcup_{T \in D} \mathbb{C}|_T, \quad \mathbb{F}|_D = \bigcup_{T \in D} \mathbb{F}|_T.$$

2.2.2. Primal Learner

The primal learning algorithm designed to learn grammars in $\mathbb{G}$ with the m-FKP is given in Algorithm 1. Here, $G_* \in \mathbb{G}$ is the target grammar. The algorithm calls the function PRIMAL to construct its conjecture.

Algorithm 1 Primal learner for $\mathbb{G}$.

```
Data: A positive presentation $T_1, T_2, \dots$ of $L_* = \mathcal{L}(G_*)$; membership oracle for $L_*$;
Result: A sequence of grammars $G_1, G_2, \dots$;
let $D_0 := \varnothing$; $K_0 := \varnothing$; $J_0 := \varnothing$; $H_0 := \varnothing$; $G_0 :=$ PRIMAL$(K_0, J_0, H_0)$;
for $i = 1, 2, \dots$ do
  let $D_i := D_{i-1} \cup \{T_i\}$; $J_i := \mathbb{C}|_{D_i}$;
  if $D_i \nsubseteq \mathcal{L}(G_{i-1})$ then
    let $K_i := \mathbb{S}|_{D_i}$; $H_i := \mathbb{F}|_{D_i}$;
  else
    let $K_i := K_{i-1}$; $H_i := H_{i-1}$;
  end if
  output $G_i :=$ PRIMAL$(K_i, J_i, H_i)$;
end for
```

Let $K \subseteq \mathbb{S}$, $J \subseteq \mathbb{C}$, $H \subseteq \mathbb{F}$ be finite sets. The grammar PRIMAL$(K, J, H) = (N, \mathbb{O}, H, \sigma, P, I) \in \mathbb{G}$ is defined as follows. Let

$$\begin{aligned} N &= \{ (\mathbf{S}, R) \in \mathscr{P}(K) \times \mathbb{R} \mid \mathbf{S} \neq \varnothing, |\mathbf{S}| \leq m, \mathbf{S} \subseteq R \}, \\ \sigma((\mathbf{S}, R)) &= R, \\ I &= \{ (\mathbf{S}, R) \in N \mid \mathbf{S} \subseteq L_* \}, \end{aligned}$$

[6]In fact, Assumption 4 can be seen to be a consequence of Assumption 6. We state Assumption 4 separately because it is often easy to check while Assumption 6 can be quite difficult to establish.

and let P consist of all rules of the form

$$(\mathbf{S}_0, R_0)(f(z_1, \ldots, z_q)) :- (\mathbf{S}_1, R_1)(z_1), \ldots, (\mathbf{S}_q, R_q)(z_q)$$

(with nonterminals in N and $f \in H$) which satisfy the following condition:

$$\forall C \in J \left(\left(R_0 \subseteq \mathrm{dom}(C) \wedge \bigwedge_{S_0 \in \mathbf{S}_0} C(S_0) \in L_* \right) \Rightarrow \bigwedge_{S_1 \in \mathbf{S}_1} \cdots \bigwedge_{S_q \in \mathbf{S}_q} C(f(S_1, \ldots, S_q)) \in L_* \right). \quad (2)$$

Note that the condition (2) is similar to the condition in part (iii) of Lemma 2.4, except that here, L_* is the language of the target grammar rather than the conjectured one and the universally quantified variable C is restricted to J (which is $\mathbb{C}|_{D_i}$ in the algorithm). (Note that the checking of (2), as well as the condition $\mathbf{S} \subseteq L_*$ in the definition of I, requires access to the membership oracle for L_*.)

The following lemma is clear from the definition of PRIMAL:

Lemma 2.6. Let $K \subseteq \mathbb{S}$, $J, J' \subseteq \mathbb{C}$, and $H \subseteq \mathbb{F}$, and let $\textsc{Primal}(K, J, H) = (N, \mathbb{O}, H, \sigma, P, I)$ and $\textsc{Primal}(K, J', H) = (N, \mathbb{O}, H, \sigma, P', I)$. We have

$$J \subseteq J' \Rightarrow P' \subseteq P.$$

We call a rule $(\mathbf{S}_0, R_0)(f(z_1, \ldots, z_q)) :- (\mathbf{S}_1, R_1)(z_1), \ldots, (\mathbf{S}_q, R_q)(z_q)$ of $\textsc{Primal}(K, J, H)$ *valid for* L_* if the following condition holds:

$$\forall C \in \mathbb{C} \left(\left(R_0 \subseteq \mathrm{dom}(C) \wedge \bigwedge_{S_0 \in \mathbf{S}_0} C(S_0) \in L_* \right) \Rightarrow \bigwedge_{S_1 \in \mathbf{S}_1} \cdots \bigwedge_{S_q \in \mathbf{S}_q} C(f(S_1, \ldots, S_q)) \in L_* \right). \quad (3)$$

This is the unrestricted version of the condition (2).

Lemma 2.7. If all rules of $G = \textsc{Primal}(K, J, H)$ are valid for L_*, then $\mathcal{L}(G) \subseteq L_*$.

Proof:
We can easily prove by induction that if ξ is an $((\mathbf{S}_1, R_1), \ldots, (\mathbf{S}_n, R_n))$-derivation environment ($n \geq 0$) of G with yield $C\colon R_1 \times \cdots \times R_n \to \mathbb{O}_0$, then $C(S_1, \ldots, S_n) \in L_*$ holds for all $S_1 \in \mathbf{S}_1, \ldots, S_n \in \mathbf{S}_n$. If τ is a complete derivation tree of G with yield S, then τ is a ()-derivation tree environment of G, so $S \in L_*$. □

Theorem 2.8. Algorithm 1 successfully learns all grammars in $\mathbb{G}$ with the m-FKP.

Proof:
Let $G_* = (N_*, \mathbb{O}, F_*, \sigma_*, P_*, I_*) \in \mathbb{G}$ be a grammar with the m-FKP. Let $\mathbf{S}_A$ be an m-kernel for each nonterminal $A \in N_*$. Without loss of generality, we may assume that for all $A \in N_*$, we have $\mathcal{S}(G_*, A) \neq \varnothing$ and $\mathcal{C}(G_*, A) \neq \varnothing$, and each $f \in F_*$ is used in some rule in P_*. This implies that all $f \in F_*$ is contained in some $T \in L_* = \mathcal{L}(G_*)$. Also, by part (i) of Lemma 2.4, each element of $\mathbf{S}_A$ is contained in some $T \in L_*$

Consider the run of Algorithm 1 on a positive presentation $T_1, T_2, \dots$ of L_*. Let k be large enough that

$$\begin{aligned} \mathbf{S}_A &\subseteq \mathbb{S}|_{D_k} \quad \text{for each } A \in N_*, \\ F_* &\subseteq \mathbb{F}|_{D_k}. \end{aligned}$$

We distinguish two cases.

Case 1. At all stages $l \geq k$, $D_l \subseteq \mathcal{L}(G_{l-1})$ holds. Then $L_* \subseteq \mathcal{L}(G_l)$ and K_l and H_l stay the same at all stages $l \geq k$. By Lemma 2.6, no new rule is added to the conjectured grammar beyond stage k. If a rule is not valid for L_*, then the context C falsifying the condition (3) must be contained in some T_j, so when $l \geq j$, J_l will contain C and the rule will not be in G_l. Hence G_l will eventually consist of valid rules only and stabilize. When that happens, by Lemma 2.7, $\mathcal{L}(G_l) \subseteq L_*$, so the conjectured grammar will be correct.

Case 2. There is some stage $l \geq k$ where $D_l \nsubseteq \mathcal{L}(G_{l-1})$. Then for all $i \geq l$, K_i will include $\mathbf{S}_A$ for all $A \in N_*$ and H_i will contain all $f \in F_*$. Then for every rule $A_0(f(z_1, \dots, z_q)) :- A_1(z_1), \dots, A_q(z_q)$ in P_*, the conjectured grammar G_i will contain the rule

$$(\mathbf{S}_{A_0}, \sigma_*(A_0))(f(z_1, \dots, z_q)) :- (\mathbf{S}_{A_1}, \sigma_*(A_1))(z_1), \dots, (\mathbf{S}_{A_q}, \sigma_*(A_q))(z_q),$$

since it is valid for L_* by part (iii) of Lemma 2.4. Also, if $A \in I_*$, $\mathbf{S}_A \subseteq L_*$ by part (ii) of Lemma 2.4, so $(\mathbf{S}_A, \sigma_*(A))$ is an initial nonterminal of G_i. This means that G_i contains a homomorphic image of G_* and $L_* \subseteq \mathcal{L}(G_i)$ holds. Similarly to Case 1, rules that are not valid for L_* will eventually be eliminated and the conjectured grammar will stabilize and be correct. □

2.2.3. Dual Learner

The dual learning algorithm designed to learn grammars in $\mathbb{G}$ with the m-FCP is given in Algorithm 2. As before, $G_* \in \mathbb{G}$ is the target grammar. The algorithm calls the function DUAL to construct its conjecture.

Algorithm 2 Dual learner for $\mathbb{G}$.

```
Data: A positive presentation $T_1, T_2, \dots$ of $L_* = \mathcal{L}(G_*)$; membership oracle for $L_*$;
Result: A sequence of grammars $G_1, G_2, \dots$;
let $D_0 := \varnothing$; $K_0 := \varnothing$; $J_0 := \varnothing$; $H_0 := \varnothing$; $G_0 :=$ DUAL$(K_0, J_0, H_0)$;
for $i = 1, 2, \dots$ do
    let $D_i := D_{i-1} \cup \{T_i\}$; $K_i := \mathbb{S}|_{D_i}$;
    if $D_i \nsubseteq \mathcal{L}(G_{i-1})$ then
        let $J_i := \mathbb{C}|_{D_i}$; $H_i := \mathbb{F}|_{D_i}$;
    else
        let $J_i := J_{i-1}$; $H_i := H_{i-1}$;
    end if
    output $G_i :=$ DUAL$(K_i, J_i, H_i)$;
end for
```

Let $K \subseteq \mathbb{S}, J \subseteq \mathbb{C}, H \subseteq \mathbb{F}$ be finite sets. The grammar DUAL$(K, J, H) = (N, \mathbb{O}, H, \sigma, P, I) \in \mathbb{G}$

is defined as follows. Let

$$N = \{ (\mathbf{C}, R) \in \mathscr{P}(J) \times \mathbb{R} \mid \mathbf{C} \neq \varnothing, |\mathbf{C}| \leq m, \mathrm{dom}(C) \subseteq R \text{ for all } C \in \mathbf{C} \},$$
$$\sigma((\mathbf{C}, R)) = R,$$
$$I = \{ (\mathbf{C}, R) \in N \mid \forall S \in K ((S \in R \wedge \bigwedge_{C \in \mathbb{C}} C(S) \in L_*) \Rightarrow S \in L_*) \},$$

and let P consist of all rules of the form

$$(\mathbf{C}_0, R_0)(f(z_1, \dots, z_q)) \mathrel{:-} (\mathbf{C}_1, R_1)(z_1), \dots, (\mathbf{C}_q, R_q)(z_q)$$

(with nonterminals in N and $f \in H$) which satisfy the following condition:

$$\forall S_1 \dots S_q \in K \left(\bigwedge_{i=1}^{q} \left(S_i \in R_i \wedge \bigwedge_{C_i \in \mathbf{C}_i} C_i(S_i) \in L_* \right) \Rightarrow \bigwedge_{C_0 \in \mathbf{C}_0} C_0(f(S_1, \dots, S_q)) \in L_* \right). \tag{4}$$

Note that the condition defining I and the condition (4) are similar to the conditions in part (ii) and (iii) of Lemma 2.5, respectively, except that here, L_* is the language of the target grammar rather than the conjectured one and the universally quantified variables $S, S_1, \dots, S_q$ are restricted to K (which is $\mathbb{S}|_{D_i}$ in the algorithm). (Checking these conditions requires access to the membership oracle for L_*.)

Again, the following property of DUAL is clear from its definition:

Lemma 2.9. Let $K, K' \subseteq \mathbb{S}$, $J \subseteq \mathbb{C}$, and $H \subseteq \mathbb{F}$, and let $\text{DUAL}(K, J, H) = (N, \mathbb{O}, H, \sigma, P, I)$ and $\text{DUAL}(K', J, H) = (N, \mathbb{O}, H, \sigma, P', I')$. We have

$$K \subseteq K' \Rightarrow (P' \subseteq P \wedge I' \subseteq I).$$

We call a rule $(\mathbf{C}_0, R_0)(f(z_1, \dots, z_q)) \mathrel{:-} (\mathbf{C}_1, R_1)(z_1), \dots, (\mathbf{C}_q, R_q)(z_q)$ of $\text{DUAL}(K, J, H)$ *valid for* L_* if the following condition holds:

$$\forall S_1 \dots S_q \in \mathbb{S} \left(\bigwedge_{i=1}^{q} \left(S_i \in R_i \wedge \bigwedge_{C_i \in \mathbf{C}_i} C_i(S_i) \in L_* \right) \Rightarrow \bigwedge_{C_0 \in \mathbf{C}_0} C_0(f(S_1, \dots, S_q)) \in L_* \right). \tag{5}$$

This is the unrestricted version of the condition (4).

We call an initial nonterminal $(\mathbf{C}, R)$ of $\text{DUAL}(K, J, H)$ *valid for* L_* if the following condition holds:

$$\forall S \in \mathbb{S} \left(\left(S \in R \wedge \bigwedge_{C \in \mathbf{C}} C(S) \in L_* \right) \Rightarrow S \in L_* \right).$$

This is the unrestricted version of the condition defining the set of initial nonterminals in $\text{DUAL}(K, J, H)$.

Lemma 2.10. If all rules and all initial nonterminals of $G = \text{DUAL}(K, J, H)$ are valid for L_*, then $\mathcal{L}(G) \subseteq L_*$.

Proof:
We can easily prove by induction that if τ is a $(\mathbf{C}, R)$-derivation tree of G with yield S, then $C(S) \in L_*$ for all $C \in \mathbf{C}$. If $(\mathbf{C}, R) \in I$, then $S \in L_*$ by the definition of validity. □

Theorem 2.11. Algorithm 2 successfully learns all grammars in $\mathbb{G}$ with the m-FCP.

Proof:
The proof is very similar to that of Theorem 2.8. Let $G_* = (N_*, \mathbb{O}, F_*, \sigma_*, P_*, I_*) \in \mathbb{G}$ be a grammar with the m-FCP. Let $\mathbf{C}_A$ be an m-context set for each nonterminal $A \in N_*$. Without loss of generality, we may assume that for all $A \in N_*$, we have $\mathcal{S}(G_*, A) \neq \varnothing$ and $\mathcal{C}(G_*, A) \neq \varnothing$, and each $f \in F_*$ is used in some rule in P_*. This implies that all $f \in F_*$ is contained in some $T \in L_*$. Also, by part (i) of Lemma 2.5, each element of $\mathbf{C}_A$ is contained in some $T \in L_*$. Let k be large enough that

$$\begin{aligned} \mathbf{C}_A &\subseteq \mathbb{C}|_{D_k} \quad \text{for each } A \in N_*, \\ F_* &\subseteq \mathbb{F}|_{D_k}. \end{aligned}$$

We distinguish two cases.

Case 1. At all stages $l \geq k$, $D_l \subseteq \mathcal{L}(G_{l-1})$ holds. Then $L_* \subseteq \mathcal{L}(G_l)$ and J_l and H_l stay the same at all stages $l \geq k$. By Lemma 2.9, no new rule is added to the conjectured grammar beyond stage k. If a rule is not valid for L_*, then the substructures $S_1, \dots, S_q$ falsifying the condition (5) must be contained in some $T_{j_1}, \dots, T_{j_q}$, so when $l \geq \max\{j_1, \dots, j_q\}$, K_l will include $\{S_1, \dots, S_q\}$ and the rule will not be in G_l. Hence G_l will eventually consist of valid rules only and stabilize. When that happens, by Lemma 2.10, $\mathcal{L}(G_l) \subseteq L_*$, so the conjectured grammar will be correct.

Case 2. There is some stage $l \geq k$ where $D_l \nsubseteq \mathcal{L}(G_{l-1})$. Then for all $i \geq l$, J_i will include $\mathbf{C}_A$ for all $A \in N_*$ and H_i will contain all $f \in F_*$. Then for every rule $A_0(f(z_1, \dots, z_q)) \mathrel{:-} A_1(z_1), \dots, A_q(z_q)$ in P_*, the conjectured grammar G_i will contain the rule

$$(\mathbf{C}_{A_0}, \sigma_*(A_0))(f(z_1, \dots, z_q)) \mathrel{:-} (\mathbf{C}_{A_1}, \sigma_*(A_1))(z_1), \dots, (\mathbf{C}_{A_q}, \sigma_*(A_q))(z_q),$$

since it is valid for L_* by part (iii) of Lemma 2.5. Also, if $A \in I_*$, the nonterminal $(\mathbf{C}_A, \sigma_*(A))$ of G_i is valid for L_* by part (ii) of Lemma 2.5 and hence is an initial nonterminal of G_i. This means that G_i contains a homomorphic image of G_*, and $L_* \subseteq \mathcal{L}(G_i)$ holds. Similarly to Case 1, rules that are not valid for L_* will eventually be eliminated and the conjectured grammar will stabilize and be correct. □

2.2.4. Efficiency of the Learning Algorithms

In the presence of the membership oracle, convergence to a correct grammar is always possible with a simple-minded "enumerative" learning algorithm that asks membership queries on the objects in $\mathbb{O}_0$ in turn and hypothesizes the first grammar in $\mathbb{G}$ that is consistent with the input data and the membership queries so far. Unlike such an enumerative algorithm, a distributional learning algorithm must be efficient both in terms of the time it takes to compute the hypothesis at each stage (*update time*) and the amount of data that guarantees convergence.

We say that a finite sequence of objects $T_1, \dots, T_n$ from a language $L \subseteq \mathbb{O}_0$ is a *locking sequence* [8] for L and a learning algorithm $\mathcal{A}$ if for some grammar G such that $\mathcal{L}(G) = L$, the algorithm $\mathcal{A}$, when given access to the oracle for L, returns G on all extensions $T_1, \dots, T_n, T'_1, \dots, T'_l$ of $T_1, \dots, T_n$ with $\{T'_1, \dots, T'_l\} \subseteq L$. If, instead, $\mathcal{A}$ outputs a grammar for a superset of L on all extensions (inside L) of $T_1, \dots, T_n$ (possibly different grammars for different extensions), then we say that $T_1, \dots, T_n$ is a *weak locking sequence* for L and $\mathcal{A}$.

Theorem 2.12. Algorithms 1 and 2 have the following properties:

(i) The update time of each algorithm is polynomial in the total size of the input data.

(ii) For each grammar $G \in \mathbb{G}$ with the m-FKP (resp. m-FCP), Algorithm 1 (resp. Algorithm 2) satisfies the following conditions:

  (a) There is a finite set $D \subseteq \mathcal{L}(G)$ whose cardinality is polynomial in the size of G such that whenever $D \subseteq \{T_1, \dots, T_n\} \subseteq \mathcal{L}(G)$, the sequence $T_1, \dots, T_n$ can be extended to a weak locking sequence for $\mathcal{L}(G)$ with the addition of at most one element of $\mathcal{L}(G)$.

  (b) For every weak locking sequence $T_1, \dots, T_n$ for $\mathcal{L}(G)$, there is a finite set E whose cardinality is polynomial in the total size of $T_1, \dots, T_n$ such that every extension $T_1, \dots, T_n, T'_1, \dots, T'_l$ of $T_1, \dots, T_n$ with $E \subseteq \{T_1, \dots, T_n, T'_1, \dots, T'_l\} \subseteq \mathcal{L}(G)$ is a locking sequence for $\mathcal{L}(G)$.

Proof:
Part (i) is by inspection of the algorithms, using Assumptions 1, 2, 3, 4, 5 and 6. The proof of part (ii) is similar to the proofs of Theorems 2.8 and 2.11. □

3. Two Learnable Classes of Nonlinear Tree Grammars

3.1. Preliminaries on Trees and Tree Patterns

A *ranked alphabet* is a finite set Δ of symbols where each symbol has a fixed rank $l \in \mathbb{N}$. We write $\Delta^{(l)}$ for the set of symbols in Δ of rank l. The set $\mathbb{T}_\Delta$ of *trees* over Δ is the smallest superset of $\Delta^{(0)}$ such that $f(T_1, \dots, T_l) \in \mathbb{T}_\Delta$ whenever $f \in \Delta^{(l)}$ and $T_1, \dots, T_l \in \mathbb{T}_\Delta$ with $l \geq 1$. The *size* $|T|$ of a tree T is the number of occurrences of symbols in it. In terms of graphical representation of trees, this is the same as the number of nodes of T.

Let X be a countably infinite set of *variables* $x_1, x_2, \dots,$ disjoint from Δ. The variables in X are assumed to have rank 0. The set of first n variables $x_1, \dots, x_n$ in X is denoted X_n. An *n-variable tree pattern* over Δ is an element of $\mathbb{T}_{\Delta \cup X_n}$ in which each of the variables $x_1, \dots, x_n$ occurs at least once. We write $\mathbb{T}_\Delta[X_n]$ for the set of n-variable tree patterns over Δ. Note that $\mathbb{T}_\Delta[X_0] = \mathbb{T}_\Delta$.

A tree $T \in \mathbb{T}_{\Delta \cup X_n}$ can naturally be viewed as a representation of a function $[\![T]\!] \colon (\mathbb{T}_{\Delta \cup X})^n \to \mathbb{T}_{\Delta \cup X}$ that maps $(U_1, \dots, U_n) \in (\mathbb{T}_{\Delta \cup X})^n$ to the tree $T[U_1, \dots, U_n]$ that results from substituting $U_1, \dots, U_n$ for $x_1, \dots, x_n$, respectively, in T. Formally, $T[U_1, \dots, U_n]$ is defined inductively by

$$\begin{aligned} a[U_1, \dots, U_n] &= a \quad \text{for } a \in \Delta^{(0)}, \\ x_i[U_1, \dots, U_n] &= U_i, \\ (f(T_1, \dots, T_l))[U_1, \dots, U_n] &= f(T_1[U_1, \dots, U_n], \dots, T_l[U_1, \dots, U_n]) \quad \text{for } f \in \Delta^{(l)}. \end{aligned}$$

Note that under this definition, $T[x_1, \dots, x_n] = T$ whenever $T \in \mathbb{T}_{\Delta \cup X_n}$.

The following should be clear from the above definition:

Lemma 3.1. For $T \in \mathbb{T}_{\Delta \cup X_n}$ and $U_1, \dots, U_n \in \mathbb{T}_{\Delta \cup X}$, $T[U_1, \dots, U_n]$ is computable in polynomial time in the combined size of $T, U_1, \dots, U_n$.

An n-variable tree pattern $Q \in \mathbb{T}_\Delta[X_n]$ is *k-copying* if each of the variables $x_1, \dots, x_n$ occurs at most k times in Q. An n-variable tree pattern Q is said to *match* a tree $T \in \mathbb{T}_\Delta$ if there are trees $T_1, \dots, T_n \in \mathbb{T}_\Delta$ such that $T = Q[T_1, \dots, T_n]$. It is easy to see that the number of k-copying n-variable tree patterns that match a tree $T \in \mathbb{T}_\Delta$ is bounded by $|T|^{kn}$ and they can be enumerated in polynomial time in $|T|$.

3.2. Enumeration of One-Variable Tree Patterns

When P_1 and P_2 are one-variable tree patterns, we call $P_1[P_2]$ the *composition* of P_1 and P_2. Note that the function $[\![P_1[P_2]]\!]$ represented by $P_1[P_2]$ is actually the composition of the functions $[\![P_1]\!]$ and $[\![P_2]\!]$. We write $\mathbb{P}_{\Delta,k}$ for the composition closure of the set of k-copying one-variable tree patterns over Δ. A key theorem we rely on in what follows is the following:

Theorem 3.2. Let $T \in \mathbb{T}_\Delta$. If $|T| = n$, then there are no more than n^{k+1} one-variable tree patterns in $\mathbb{P}_{\Delta,k}$ that match T, and they can be enumerated in polynomial time.

The proof of this theorem is given in the long appendix (Section A).

3.3. Parallel Regular Tree Grammars

A *parallel regular tree grammar* is a nonlinear variant of the well-known regular tree grammar. Here, the set $\mathbb{T}_\Delta$ plays the role of $\mathbb{O} = \mathbb{O}_0$, and $\mathbb{R}$ is the singleton of $\mathbb{T}_\Delta$. Functions used in a parallel regular tree grammar are ones that are represented by tree patterns.

Example 3.3. Let $\Delta = \Delta^{(0)} \cup \Delta^{(1)} \cup \Delta^{(2)} \cup \Delta^{(4)} = \{a\} \cup \{g\} \cup \{f\} \cup \{h\}$. Define a generalized context-free grammar $G = (N, O, F, \sigma, P, I)$, where

$$\begin{aligned}
N &= \{A_1, A_2, A_3\},\\
O &= \mathbb{T}_\Delta,\\
F &= F_0 \cup F_1 \cup F_2,\\
F_0 &= \{a\},\\
F_1 &= \{[\![f(x_1)]\!], [\![g(g(x_1))]\!]\},\\
F_2 &= \{[\![h(x_1, x_2, x_1, x_2)]\!]\},\\
\sigma(Z) &= \mathbb{T}_\Delta \quad \text{for all } Z \in N,\\
I &= \{A_3\},
\end{aligned}$$

and P consists of the following rules:

$$\begin{aligned}
A_1(a) &:- \,,\\
A_1(f(z_1, z_1))) &:- A(z_1),\\
A_2(a) &:- \,,\\
A_2(g(g(z_1)) &:- A_2(z_1),\\
A_3(h(z_1, z_2, z_1, z_2)) &:- A_1(z_1), A_2(z_2).
\end{aligned}$$

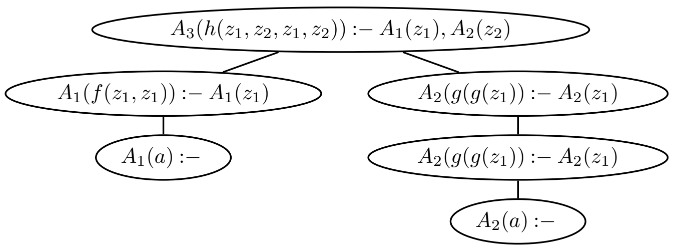
$A_3(h(z_1, z_2, z_1, z_2)) :- A_1(z_1), A_2(z_2)$
$A_1(f(z_1, z_1)) :- A_1(z_1)$
$A_2(g(g(z_1)) :- A_2(z_1)$
$A_1(a) :-$
$A_2(g(g(z_1)) :- A_2(z_1)$
$A_2(a) :-$

Figure 1. A complete derivation tree of a parallel regular tree grammar.

(Note that we are expanding $[\![f(x_1, x_1)]\!](z_1)$ as $f(z_1, z_1)$, etc.) This is an example of a parallel regular tree grammar. Figure 1 shows an example of a complete derivation tree of this grammar. The yield of this derivation tree is $h(f(a, a), g(g(g(g(a)))), f(a, a), g(g(g(g(a)))))$.

Since the value of the sort function σ in a parallel regular tree grammar is always $\mathbb{T}_\Delta$, it is superfluous. So a parallel regular tree grammar $(N, \mathbb{T}_\Delta, F, \sigma, P, I)$ may be thought of as a 5-tuple $(N, \mathbb{T}_\Delta, F, P, I)$.

To satisfy the requirements about the target grammar class, we pick arbitrary positive integers k, r and limit functions in grammars to those represented by k-copying q-variable tree patterns with $q \leq r$. Let

$$\begin{aligned}
\mathbb{O} &= \mathbb{T}_\Delta, \\
\mathbb{F}_q &= \{\, [\![Q]\!] \mid Q \text{ is a } k\text{-copying } q\text{-variable tree pattern} \,\}, \\
\mathbb{F} &= \bigcup_{q=0}^{r} \mathbb{F}_q.
\end{aligned}$$

This determines the grammar class

$$\mathbb{G}_{\mathrm{PRTG}(k,r)} = \{\, (N, \mathbb{O}, F, P, I) \mid N \subseteq \mathbb{N}, F \subseteq \mathbb{F} \,\}.$$

The grammar in Example 3.3 belongs to $\mathbb{G}_{\mathrm{PRTG}(k,r)}$ if $k \geq 2$ and $r \geq 2$.

Let $\mathbb{S}_{\mathrm{PRTG}(k,r)}$ and $\mathbb{C}_{\mathrm{PRTG}(k,r)}$ be the sets of substructures and contexts of $\mathbb{G}_{\mathrm{PRTG}(k,r)}$, respectively. Write $\mathbb{F}_{\mathrm{PRTG}(k,r)}$ for the set $\mathbb{F}$ defined just above.

It is easy to see that $\mathbb{S}_{\mathrm{PRTG}(k,r)} = \mathbb{T}_\Delta$. The following lemma is also straightforward:

Lemma 3.4. $\mathbb{C}_{\mathrm{PRTG}(k,r)} = \{\, [\![Q]\!] \mid Q \in \mathbb{P}_{\Delta,k} \,\}$.

Let us check that the conditions required for Algorithms 1 and 2 to work correctly are satisfied. Assumptions 1 and 2 are trivial. The first two items of Assumption 3 are trivial and the third item holds by Lemma 3.1. Assumption 4 can be shown to hold by a standard dynamic programming technique. (Here, the bound r on the number of occurrences of nonterminals on the right-hand side of rules plays an important role.) The first two items of Assumption 5 are trivial, and the third item follows from Lemma 3.1. As for Assumption 6, since $\mathbb{S}_{\mathrm{PRTG}(k,r)} = \mathbb{T}_\Delta$ implies that $\mathbb{S}_{\mathrm{PRTG}(k,r)}|_T$ is the set of subtrees of T, the polynomial-time enumerability of $\mathbb{S}_{\mathrm{PRTG}(k,r)}|_T$ is trivial. The polynomial-time enumerability of $\mathbb{C}_{\mathrm{PRTG}(k,r)}|_T$ directly follows from Lemma 3.4 and Theorem 3.2. The elements of $\mathbb{F}_{\mathrm{PRTG}(k,r)}|_T$ are the k-copying q-variable tree patterns ($q \leq r$) that match some subtree of T, so this set is again clearly enumerable in polynomial time in the size of T.

Theorem 3.5. Algorithms 1 and 2, when applied to $\mathbb{G} = \mathbb{G}_{\mathrm{PRTG}(k,r)}$, successfully learn all grammars in $\mathbb{G}_{\mathrm{PRTG}(k,r)}$ with the m-FKP and the m-FCP, respectively, while satisfying favorable properties in terms of efficiency.

3.4. Uniformly Copying IO Context-Free Tree Grammars

An *IO context-free tree grammar* [3] is a generalized context-free grammar where each object is an n-variable tree pattern for some $n \geq 0$.[7]

Each function in an IO context-free tree grammar is represented by a tree pattern over an extended ranked alphabet $\Delta \cup Y$, where Y is a ranked alphabet of *second-order variables*, disjoint from Δ. We assume that for each $n \geq 0$, $Y^{(n)} = \{y_1^{(n)}, y_2^{(n)}, \dots\}$. A second-order variable $y_i^{(n)} \in Y^{(n)}$ is interpreted as ranging over the functions represented by elements of $\mathbb{T}_\Delta[X_n]$. We adopt the convention according to which $y_i^{(n)}(T_1, \dots, T_n)$ denotes $y_i^{(0)}$ when $n = 0$.

Let $Q \in \mathbb{T}_{\Delta \cup Y}[X_n]$ be an n-variable tree pattern over $\Delta \cup Y$ such that the symbols from Y occurring in Q are $y_1^{(n_1)}, \dots, y_q^{(n_q)}$. The function $\langle\!\langle Q \rangle\!\rangle$ that Q represents maps $(U_1, \dots, U_q) \in \mathbb{T}_\Delta[X_{n_1}] \times \dots \times \mathbb{T}_\Delta[X_{n_q}]$ to the n-variable tree pattern $Q\langle U_1, \dots, U_q \rangle$ that is obtained by interpreting each $y_i^{(n_i)}$ in Q as $[\![U_i]\!]$. We inductively define $T\langle U_1, \dots, U_q \rangle$ for $T \in \mathbb{T}_{\Delta \cup \{y_1^{(n_1)}, \dots, y_q^{(n_q)}\} \cup X_n}$ and $U_i \in \mathbb{T}_\Delta[X_{n_i}]$ $(i = 1, \dots, q)$ as follows:

$$\begin{aligned}
a\langle U_1, \dots, U_q \rangle &= a \quad \text{for } a \in \Delta^{(0)}, \\
x_i\langle U_1, \dots, U_q \rangle &= x_i, \\
(f(T_1, \dots, T_l))\langle U_1, \dots, U_q \rangle &= f(T_1\langle U_1, \dots, U_q \rangle, \dots, T_l\langle U_1, \dots, U_q \rangle) \quad \text{for } f \in \Delta^{(l)}, \\
(y_i^{(n_i)}(T_1, \dots, T_{n_i}))\langle U_1, \dots, U_q \rangle &= U_i[T_1\langle U_1, \dots, U_q \rangle, \dots, T_{n_i}\langle U_1, \dots, U_q \rangle].
\end{aligned}$$

Let $\mathbb{T}_\Delta\langle Y_q \rangle[X_n]$ denote the set of n-variable tree patterns $Q \in \mathbb{T}_{\Delta \cup Y}[X_n]$ such that the symbols from Y occurring in Q are $y_1^{(n_1)}, \dots, y_q^{(n_q)}$ for some $n_1, \dots, n_q$ and for each $i = 1, \dots, q$, $y_i^{(n_i)}$ occurs exactly once in Q. A function used by an IO context-free tree grammar is $\langle\!\langle Q \rangle\!\rangle$ for some $Q \in \bigcup_{q \in \mathbb{N}} \bigcup_{n \in \mathbb{N}} \mathbb{T}_\Delta\langle Y_q \rangle[X_n]$. In standard formulation, the sort of a nonterminal is $\mathbb{T}_\Delta[X_n]$ for some n, but we will consider a more general formulation which expedites the definition of what we call a *uniformly copying IO context-free tree grammar*.

Example 3.6. Let $\Delta = \Delta_0 \cup \Delta_2 = \{a\} \cup \{f\}$. Consider a generalized context-free grammar $G = (N, O, F, \sigma, P, I)$, where

$$\begin{aligned}
N &= \{A_0, A_1, A_2, A_3\}, \\
O &= \mathbb{T}_\Delta \cup \mathbb{T}_\Delta[X_1], \\
F &= F_0 \cup F_1 \cup F_2,
\end{aligned}$$

[7] What we are calling an IO context-free tree grammar is actually a *non-deleting* IO context-free tree grammar. (This restriction does not affect the generated languages.) "IO" stands for the *inside-out* mode of rewriting, where an innermost nonterminal in a sentential form is rewritten at each step. This mode of rewriting can be cast as bottom-up derivation. An *outside-in* or *OI* context-free tree grammar [3] does not have an associated notion of derivation tree and hence does not count as a generalized context-free grammar. The IO and OI modes of rewriting were originally introduced by Fischer [4] in his study of *macro grammars*.

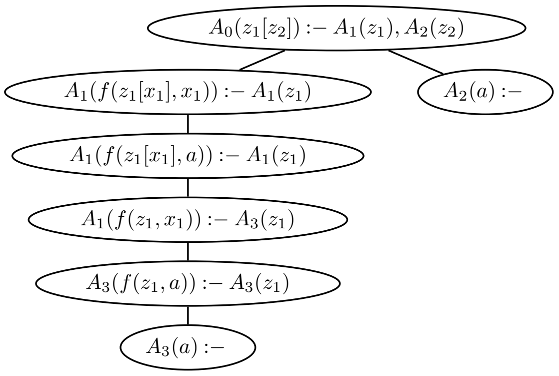
$A_0(z_1[z_2]) :- A_1(z_1), A_2(z_2)$
$A_1(f(z_1[x_1], x_1)) :- A_1(z_1)$
$A_2(a) :-$
$A_1(f(z_1[x_1], a)) :- A_1(z_1)$
$A_1(f(z_1, x_1)) :- A_3(z_1)$
$A_3(f(z_1, a)) :- A_3(z_1)$
$A_3(a) :-$

Figure 2. An example of a complete derivation tree of an IO context-free tree grammar.

$$\begin{aligned}
F_0 &= \{x_1, a\},\\
F_1 &= \{\langle\!\langle f(y_1^{(1)}(x_1), x_1)\rangle\!\rangle, \langle\!\langle f(y_1^{(1)}(x_1), a)\rangle\!\rangle, \langle\!\langle f(y_1^{(1)}(x_1), a)\rangle\!\rangle, \langle\!\langle f(y_1^{(0)}, a)\rangle\!\rangle\},\\
F_2 &= \{\langle\!\langle y_1^{(1)}(y_2^{(0)})\rangle\!\rangle\},\\
\sigma(A_0) &= \mathbb{T}_\Delta,\\
\sigma(A_1) &= \mathbb{T}_\Delta[X_1],\\
\sigma(A_2) &= \mathbb{T}_\Delta,\\
\sigma(A_3) &= \mathbb{T}_\Delta,\\
I &= \{A_0\},
\end{aligned}$$

and P consists of the following rules:

$$\begin{aligned}
A_0(z_1[z_2]) &:- A_1(z_1), A_2(z_2),\\
A_1(x_1) &:- ,\\
A_1(f(z_1[x_1], x_1)) &:- A_1(z_1),\\
A_1(f(z_1[x_1], a)) &:- A_1(z_1),\\
A_1(f(z_1, x)) &:- A_3(z_1),\\
A_2(a) &:- ,\\
A_3(a) &:- ,\\
A_3(f(z_1, a)) &:- A_3(z_1).
\end{aligned}$$

(Note that for example, $\langle\!\langle f(y_1^{(1)}(x_1), x_1)\rangle\!\rangle(T) = f(T[x_1], x_1)$ for every $T \in \mathbb{T}_\Delta[X_1]$, so we are writing $f(z_1[x_1], x_1)$ for $\langle\!\langle f(y_1^{(1)}(x_1), x_1)\rangle\!\rangle(z_1)$, etc.) This is an example of an IO context-free tree grammar. Figure 2 shows an example of a complete derivation tree of this grammar. The yield of this derivation tree is $f(f(f(f(a, a), a), a), a)$.

The language of this grammar consists of all purely left-branching trees over Δ, i.e., $\{a, f(a, a), f(f(a, a), a), f(f(f(a, a), a), a), \dots\}$. The grammar is designed in such a way that $\mathcal{S}(G, A_1)$ and $\mathcal{C}(G, A_2)$ both contain all one-variable tree patterns over Δ that match any element of $\mathcal{L}(G)$.

As before, an n-variable tree pattern $Q \in \mathbb{T}_{\Delta \cup Y}[X_n]$ is said to be *k-copying* if each x_i occurs at most k times in it. Pick arbitrary positive integers k, p, r and for $q = 1, \dots, r$, let $\mathbb{Q}_q$ be the set of k-copying tree patterns $Q \in \bigcup_{n=0}^{p} \mathbb{T}_\Delta\langle Y_q\rangle[X_n]$ such that the second-order variables in Q are $y_1^{(n_1)}, \dots, y_q^{(n_q)}$ with $n_i \le p$. Let

$$\begin{aligned}
\mathbb{O} &= \bigcup_{n=0}^{p} \mathbb{T}_\Delta[X_n],\\
\mathbb{O}_0 &= \mathbb{T}_\Delta,\\
\mathbb{F}_q &= \{\, \langle\!\langle Q \rangle\!\rangle \mid Q \in \mathbb{Q}_q \,\},\\
\mathbb{F} &= \bigcup_{q=0}^{r} \mathbb{F}_q,\\
\mathbb{R} &= \{\, \mathbb{T}_\Delta[X_n] \mid 0 \le n \le p \,\}.
\end{aligned}$$

This determines the grammar class

$$\mathbb{G}_{\mathrm{IOCFTG}(k,p,r)} = \{\, (N, \mathbb{O}, F, \sigma, P, I) \mid N \subseteq \mathbb{N}, F \subseteq \mathbb{F}, \mathrm{ran}(\sigma) \subseteq \mathbb{R}, \sigma(I) \subseteq \mathscr{P}(\mathbb{O}_0) \,\}.$$

(Since $\mathbb{R} \cap \mathscr{P}(\mathbb{O}_0) = \{\mathbb{T}_\Delta\}$, grammars in this class must satisfy $\sigma(A) = \mathbb{T}_\Delta$ for all $A \in I$.)

The grammar G in Example 3.6 belongs to $\mathbb{G}_{\mathrm{IOCFTG}(k,p,r)}$ if $k \ge 2$ and $r \ge 2$. This example shows that the class does not satisfy Assumption 6 because when $T \in \mathcal{L}(G)$ has n occurrences of a, there are at least $2^n - 1$ A_1-substructures and exactly $2^n - 1$ A_2-contexts that are contained in T, which means that neither the substructures nor the contexts contained in T can be enumerated explicitly in polynomial time in the size of T.

In order to define a subclass of the IO context-free tree grammars to which Algorithms 1 and 2 are applicable, we have to impose some qualitative condition on the possible rules of the grammars. For this purpose, we use a more fine-grained classification of objects into sorts. Different sorts may overlap, so the set of sorts will no longer be a partition of the set of objects.

For an n-variable tree pattern $T \in \mathbb{T}_\Delta[X_n]$, its *duplicity vector* is the bit vector $d(T) \in \{0,1\}^n$ whose ith bit is 1 if and only if x_i occurs more than once in T. We assume that when $T \in \mathbb{T}_\Delta[X_0] = \mathbb{T}_\Delta$, the duplicity vector of T is always $()$, the "vector" with no component. We use the set $\{0,1\}^n$ together with the familiar "component-wise" partial order $\le$; in other words, we view $\{0,1\}^n$ as the n-fold product of the ordered set $(\{0,1\}, \le)$. If $\vec{v} \in \{0,1\}^n$, we let

$$R_{\vec{v}} = \{\, T \in \mathbb{T}_\Delta[X_n] \mid d(T) \le \vec{v} \,\}.$$

Note that $R_{(1,\dots,1)} = \mathbb{T}_\Delta[X_n]$, where n is the length of the vector $(1, \dots, 1)$. In particular, $R_{()} = \mathbb{T}_\Delta$.

Let $Q \in \mathbb{T}_\Delta\langle Y_q\rangle[X_n]$ be an n-variable tree pattern whose second-order variables are $y_1^{(n_1)}, \dots, y_q^{(n_q)}$, and for each $i = 1, \dots, q$, let $U_i \in R_{\vec{v_i}}$ for some $\vec{v_i} \in \{0,1\}^{n_i}$. Then $\langle\!\langle Q \rangle\!\rangle(U_1, \dots, U_q) = Q\langle U_1, \dots, U_q\rangle \in \mathbb{T}_\Delta[X_n]$. For each $j = 1, \dots, n$, it is clear that x_j occurs more times in $Q\langle U_1, \dots, U_q\rangle$ than in Q if and only if there exist i and k such that the kth argument of the unique occurrence of $y_i^{(n_i)}$ in Q contains an an occurrence of x_j, and x_k occurs more than once in U_i. Note that this is only possible if the kth bit of $\vec{v_i}$ is 1.

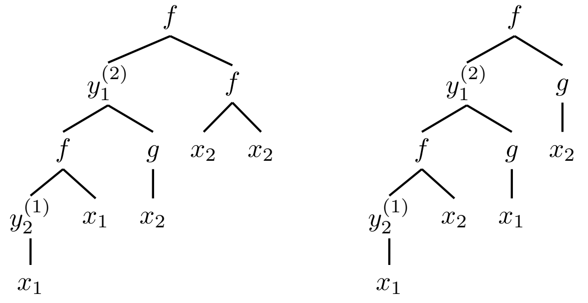

Figure 3. A 3-copying 2-variable tree pattern that is uniformly copying on $R_{(1,0)} \times R_{(0)}$ (left) and a 2-copying 2-variable tree pattern that is not (right).

Definition 3.7. Let $Q \in \mathbb{T}_\Delta\langle Y_q\rangle[X_n]$ be an n-variable tree pattern whose second-order variables are $y_1^{(n_1)}, \dots, y_q^{(n_q)}$. Let $\vec{v_i} \in \{0,1\}^{n_i}$ for each $i = 1, \dots, q$. We say that Q is *uniformly copying* on $R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}}$ if the occurrences of $x_1, \dots, x_n$ in Q satisfy the following condition:

- whenever there is an occurrence of x_j inside the kth argument of $y_i^{(n_i)}$ such that the kth bit of $\vec{v_i}$ is 1, all occurrences of x_j are inside the kth argument of $y_i^{(n_i)}$.

Example 3.8. Consider the following tree patterns in $\mathbb{T}_\Delta\langle Y_2\rangle[X_2]$:

$$f(y_1^{(2)}(f(y_2^{(1)}(x_1), x_1), g(x_2)), f(x_2, x_2)), \quad f(y_1^{(2)}(f(y_2^{(1)}(x_1), x_2), g(x_1)), g(x_2)).$$

Figure 3 shows these tree patterns in graphical notation. The tree pattern on the left is uniformly copying on $R_{(1,0)} \times R_{(0)}$, because all occurrences of x_1 are inside the first argument of $y_1^{(2)}$ and all occurrences of x_2 are outside it. The tree pattern on the right is not uniformly copying on $R_{(1,0)} \times R_{(0)}$. Both variables x_1 and x_2 have two occurrences, one inside the first argument of $y_1^{(2)}$ and one outside it, violating the condition for uniform copying.

When $f\colon \mathbb{T}_\Delta[X_{n_1}] \times \cdots \times \mathbb{T}_\Delta[X_{n_q}] \to \mathbb{T}_\Delta[X_n]$ is a q-ary function and $\vec{v_i} \in \{0,1\}^{n_i}$ for $i = 1, \dots, q$, we let

$$f_{\vec{v_1},\dots,\vec{v_q}} = f \restriction (R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}}),$$

the restriction of f to $R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}}$.

Now pick arbitrary positive integers k, p, r. Let

$$\begin{aligned} \mathbb{F}'_q = \{\, \langle\!\langle Q \rangle\!\rangle_{\vec{v_1},\dots,\vec{v_q}} \mid {} & Q \in \bigcup_{n=0}^{p} \mathbb{T}_\Delta\langle Y_q\rangle[X_n], Q \text{ is } k\text{-copying}, \\ & \text{the second-order variables in } Q \text{ are } y_1^{(n_1)}, \dots, y_q^{(n_q)}, \\ & n_i \le p \text{ and } \vec{v_i} \in \{0,1\}^{n_i} \text{ for } i = 1, \dots, q, \\ & Q \text{ is uniformly copying on } R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}} \,\}, \end{aligned}$$

$$\mathbb{F}' = \bigcup_{q=0}^{r} \mathbb{F}'_q,$$

$$\mathbb{R}' = \{ R_{\vec{v}} \mid \vec{v} \in \bigcup_{n=0}^{p} \{0,1\}^n \}.$$

This determines the grammar class

$$\mathbb{G}_{\text{UC-IOCFTG}(k,p,r)} = \{ (N, \mathbb{O}, F, \sigma, P, I) \mid N \subseteq \mathbb{N}, F \subseteq \mathbb{F}', \operatorname{ran}(\sigma) \subseteq \mathbb{R}', \sigma(I) \subseteq \mathscr{P}(\mathbb{O}_0) \},$$

where $\mathbb{O} = \bigcup_{n=0}^{p} \mathbb{T}_\Delta[X_n]$ and $\mathbb{O}_0 = \mathbb{T}_\Delta = R_{()}$ as before. (Since $\mathbb{R}' \cap \mathscr{P}(\mathbb{O}_0) = \{\mathbb{T}_\Delta\} = \{R_{()}\}$, grammars in this class must satisfy $\sigma(A) = \mathbb{T}_\Delta$ for all $A \in I$.) We call a grammar in this class a *uniformly copying IO context-free tree grammar*.

Example 3.9. Let $\Delta = \Delta_0 \cup \Delta_1 \cup \Delta_2 = \{a\} \cup \{g\} \cup \{f\}$. Consider the uniformly copying IO context-free tree grammar $G = (N, \mathbb{O}, F, \sigma, P, I) \in \mathbb{G}_{\text{UC-IOCFTG}(2,2,2)}$, where

$$\begin{aligned}
N &= \{A_0, A_1, A_2\},\\
F &= F_0 \cup F_1 \cup F_2,\\
F_0 &= \{f(x_1, x_2), x_1\},\\
F_1 &= \{\langle\!\langle y_1^{(2)}(a, a)\rangle\!\rangle_{(1,0)}, \langle\!\langle g(y_1^{(1)}(x_1))\rangle\!\rangle_{(0)}\},\\
F_2 &= \{\langle\!\langle y_1^{(2)}(f(y_2^{(1)}(x_1), x_1), x_2)\rangle\!\rangle_{(1,0),(0)}\},\\
\sigma(A_0) &= R_{()} = \mathbb{T}_\Delta,\\
\sigma(A_1) &= R_{(0)},\\
\sigma(A_2) &= R_{(1,0)},\\
I &= \{A_0\},
\end{aligned}$$

and P consists of the following rules:

$$\begin{aligned}
A_0(z[a, a]) &:- A_2(z_1),\\
A_2(f(x_1, x_2)) &:- ,\\
A_2(z_1[f(z_2[x_1], x_1), x_2]) &:- A_2(z_1), A_1(z_2),\\
A_1(x_1) &:- ,\\
A_1(g(z_1[x_1])) &:- A_1(z_1).
\end{aligned}$$

Note that $y_1^{(2)}(f(y_2^{(1)}(x_1), x_1), x_2)$ is uniformly copying on $R_{(1,0)} \times R_{(0)}$. Figure 4 shows an example of a complete derivation tree of this grammar. The yield of this derivation tree is $f(f(g(f(g(a), a)), f(g(a), a)), a))$.

Let $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}$ and $\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$ be the sets of substructures and contexts of $\mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$, respectively, and write $\mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}$ for the set $\mathbb{F}'$ defined above.

For $n \geq 0$, define

$$\begin{aligned}
\mathbb{P}_{\Delta,k}^{(n)} = \{ U \in \mathbb{T}_\Delta[X_n] \mid\ &\text{for } j = 1, \dots, n,\\
&U[T_1, \dots, T_{j-1}, x_1, T_{j+1}, \dots, T_n] \in \mathbb{P}_{\Delta,k}\\
&\text{whenever } T_1, \dots, T_{j-1}, T_{j+1}, \dots, T_n \in \mathbb{T}_\Delta \}.
\end{aligned}$$

Note that $\mathbb{P}_{\Delta,k}^{(0)} = \mathbb{T}_\Delta$ and $\mathbb{P}_{\Delta,k}^{(1)} = \mathbb{P}_{\Delta,k}$.

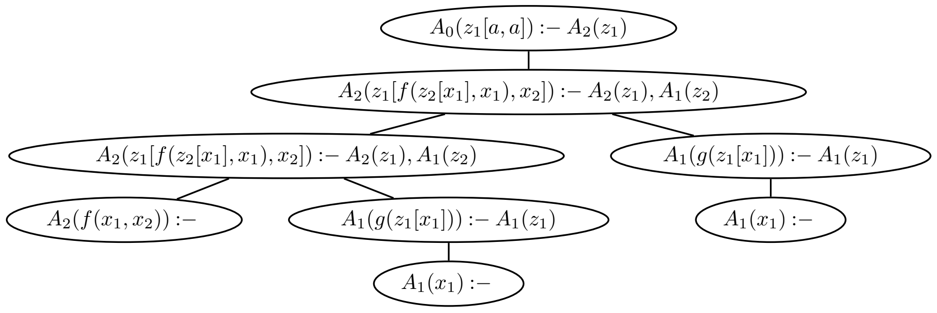
$A_0(z_1[a,a]) :- A_2(z_1)$
$A_2(z_1[f(z_2[x_1],x_1),x_2]) :- A_2(z_1), A_1(z_2)$
$A_2(z_1[f(z_2[x_1],x_1),x_2]) :- A_2(z_1), A_1(z_2)$
$A_1(g(z_1[x_1])) :- A_1(z_1)$
$A_2(f(x_1,x_2)) :-$
$A_1(g(z_1[x_1])) :- A_1(z_1)$
$A_1(x_1) :-$
$A_1(x_1) :-$

Figure 4. A complete derivation tree of a uniformly copying IO context-free tree grammar.

Lemma 3.10. Let $Q \in \mathbb{T}_\Delta\langle Y_q\rangle[X_1]$ be a 1-copying one-variable tree pattern with second-order variables $y_1^{(n_1)}, \dots, y_q^{(n_q)}$. If $U_i \in \mathbb{P}_{\Delta,k}^{(n_i)}$ for $i = 1, \dots, q$, then $Q\langle U_1, \dots, U_q\rangle \in \mathbb{P}_{\Delta,k}$.

Proof:
We show by induction that $\overline{T} = T\langle U_1, \dots, U_q\rangle$ belongs to $\mathbb{P}_{\Delta,k}$ for every subtree T of Q that contains the unique occurrence of x_1 in Q.

If $T = x_1$, then $\overline{T} = x_1 \in \mathbb{P}_{\Delta,k}$.

If $T = f(T'_1, \dots, T'_l)$ and T'_i contains the occurrence of x_1, then $\overline{T} = f(\overline{T'_1}, \dots, \overline{T'_l})$ is the composition of $f(\overline{T'_1}, \dots, \overline{T'_{i-1}}, x_1, \overline{T'_{i+1}}, \dots, \overline{T'_l})$ and $\overline{T'_i}$. The former is a 1-copying one-variable tree pattern and the latter belongs to $\mathbb{P}_{\Delta,k}$ by induction hypothesis, so $\overline{T}$ belongs to $\mathbb{P}_{\Delta,k}$.

If $T = y_h^{(n_h)}(T'_1, \dots, T'_{n_h})$ and T'_i contains the occurrence of x_1, then $\overline{T} = U_h[\overline{T'_1}, \dots, \overline{T'_{n_h}}]$ is the composition of $U_h[\overline{T'_1}, \dots, \overline{T'_{i-1}}, x_1, \overline{T'_{i+1}}, \dots, \overline{T'_l}]$ and $\overline{T'_i}$. The former belongs to $\mathbb{P}_{\Delta,k}$ since $U_h \in \mathbb{P}_{\Delta,k}^{(n_h)}$ and $\overline{T_j} \in \mathbb{T}_\Delta$ for $j = 1, \dots, i-1, i+1, \dots, l$, and the latter belongs to $\mathbb{P}_{\Delta,k}$ by induction hypothesis. Therefore, $\overline{T}$ belongs to $\mathbb{P}_{\Delta,k}$. □

The definition of "uniformly copying" is designed to make the following lemma hold:

Lemma 3.11. Let $Q \in \mathbb{T}_\Delta\langle Y_q\rangle[X_n]$ be a k-copying n-variable tree pattern with second-order variables $y_1^{(n_1)}, \dots, y_q^{(n_q)}$ and let $\vec{v_i} \in \{0,1\}^{n_i}$. If Q is uniformly copying on $R_{\vec{v_1}} \times \dots \times R_{\vec{v_q}}$ and $U_i \in \mathbb{P}_{\Delta,k}^{(n_i)} \cap R_{\vec{v_i}}$ for each $i = 1, \dots, n$, then $Q\langle U_1, \dots, U_q\rangle \in \mathbb{P}_{\Delta,k}^{(n)}$.

Proof:
Let $T_1, \dots, T_n \in \mathbb{T}_\Delta$. It suffices to show

$$\overline{T} = T\langle U_1, \dots, U_q\rangle[T_1, \dots, T_{j-1}, x_1, T_{j+1}, \dots, T_n] \in \mathbb{P}_{\Delta,k}$$

for each subtree T of Q that contains an occurrence of x_j. We do so by induction on T.

Case 1. $T = x_j$. We have

$$\overline{T} = x_j\langle U_1, \dots, U_q\rangle[T_1, \dots, T_{j-1}, x_1, T_{j+1}, \dots, T_n] = x_1 \in \mathbb{P}_{\Delta,k}.$$

Case 2. $T = f(Q_1, \dots, Q_l)$, $f \in \Delta^{(l)}$. We distinguish two cases depending on the cardinality of $\{ m \mid x_j \text{ occurs in } Q_m \}$.

Case 2.1. There is a unique m such that x_j occurs in Q_m. Then $\overline{Q_{m'}} \in \mathbb{T}_\Delta$ for $m' \neq m$ and $\overline{T} = f(\overline{Q_1}, \ldots, \overline{Q_l})$ is the composition of $f(\overline{Q_1}, \ldots, \overline{Q_{m-1}}, x_1, \overline{Q_{m+1}}, \ldots, \overline{Q_l})$ and $\overline{Q_m}$. The former is a 1-copying one-variable tree pattern over Δ, and the latter belongs to $\mathbb{P}_{\Delta,k}$ by induction hypothesis. Therefore, $\overline{T} \in \mathbb{P}_{\Delta,k}$.

Case 2.2. There are m, m' such that $m \neq m'$ and x_j occurs in Q_m and in $Q_{m'}$. Since $Q \in \mathbb{T}\langle Y_q \rangle[X_n]$ is uniformly copying on $R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}}$, this means that there are no i, k such that x_j occurs in $T = f(Q_1, \ldots, Q_l)$ inside the kth argument of $y_i^{(n_i)}$ and the kth bit of $\vec{v_i}$ is 1. (Recall that each $y_i^{(n_i)}$ occurs exactly once in Q.) It follows that the number of occurrences of x_j in $T\langle U_1, \ldots, U_q \rangle$ is the same as the number of occurrences of x_j in T, which is $\leq k$ since Q is k-copying. Therefore, x_1 occurs at most k times in $\overline{T} = T\langle U_1, \ldots, U_q \rangle[T_1, \ldots, T_{j-1}, x_1, T_{j+1}, \ldots, T_n]$ and hence $\overline{T} \in \mathbb{P}_{\Delta,k}$.

Case 3. $T = y_h^{(n_h)}(Q_1, \ldots, Q_{n_h})$. Again, we distinguish two cases depending on the cardinality of $\{ m \mid x_j \text{ occurs in } Q_m \}$.

Case 3.1. There is a unique m such that x_j occurs in Q_m. Then $\overline{Q_{m'}} \in \mathbb{T}_\Delta$ for $m' \neq m$ and $\overline{T} = U_h[\overline{Q_1}, \ldots, \overline{Q_{n_h}}]$ is the composition of $U_h[\overline{Q_1}, \ldots, \overline{Q_{m-1}}, x_1, \overline{Q_{m+1}}, \ldots, \overline{Q_{n_h}}]$ and $\overline{Q_m}$. Since $U_h \in \mathbb{P}_{\Delta,k}^{(n_h)}$, we have $U_h[\overline{Q_1}, \ldots, \overline{Q_{m-1}}, x_1, \overline{Q_{m+1}}, \ldots, \overline{Q_{n_h}}] \in \mathbb{P}_{\Delta,k}$. Since x_j occurs in Q_m, we also get $\overline{Q_m} \in \mathbb{P}_{\Delta,k}$ by induction hypothesis. Therefore, $\overline{T} \in \mathbb{P}_{\Delta,k}$.

Case 3.2. There are m, m' such that $m \neq m'$ and x_j occurs in Q_m and in $Q_{m'}$. Since $Q \in \mathbb{T}\langle Y_q \rangle[X_n]$ is uniformly copying, this means that there are no i, k such that x_j occurs in $T = y_h^{(n_h)}(Q_1, \ldots, Q_{n_h})$ inside the kth argument of $y_i^{(n_i)}$ and the kth bit of $\vec{v_i}$ is 1. Similarly to Case 2.2, it follows that x_j occurs exactly the same number of times in $T\langle U_1, \ldots, U_q \rangle$ as it does in T, which is $\leq k$. Therefore, x_1 occurs at most k times in $\overline{T} = T\langle U_1, \ldots, U_q \rangle[T_1, \ldots, T_{j-1}, x_1, T_{j+1}, \ldots, T_n]$ and $\overline{T} \in \mathbb{P}_{\Delta,k}$. □

Lemma 3.12.

$$\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)} \subseteq \bigcup_{n=0}^{p} \mathbb{P}_{\Delta,k}^{(n)}.$$

Proof:
Let $G = (N, \mathbb{O}, F, \sigma, P, I) \in \mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$. We show that $\mathcal{S}(G, A) \subseteq \mathbb{P}_{\Delta,k}^{(n)}$ for each $A \in N^{(n)}$ by induction on A-derivation trees. Let

$$A(\langle\!\langle Q \rangle\!\rangle_{\vec{v_1},\ldots,\vec{v_q}}(z_1, \ldots, z_q)) :- A_1(z_1), \ldots, A_q(z_q)$$

be a rule in P, where $\sigma(A) = R_{\vec{v}} \subseteq \mathbb{T}_\Delta[X_n]$ and $\sigma(A_i) \subseteq R_{\vec{v_i}} \subseteq \mathbb{T}_\Delta[X_{n_i}]$. Assume that for $i = 1, \ldots, q$, $U_i \in \mathcal{S}(G, A_i)$. Then $U_i \in R_{\vec{v_i}}$, and by induction hypothesis, $U_i \in \mathbb{P}_{\Delta,k}^{(n_i)}$. Since $\langle\!\langle Q \rangle\!\rangle_{\vec{v_1},\ldots,\vec{v_q}} \in \mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}$, Q is k-copying and uniformly copying on $R_{\vec{v_1}} \times \cdots \times R_{\vec{v_q}}$. By Lemma 3.11, we get $\langle\!\langle Q \rangle\!\rangle_{\vec{v_1},\ldots,\vec{v_q}}(U_1, \ldots, U_q) = Q\langle U_1, \ldots, U_q \rangle \in \mathbb{P}_{\Delta,k}^{(n)}$. □

The converse inclusion does not seem to hold, and currently we do not have an exact characterization of $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}$. However, Lemma 3.12 alone is sufficient to establish that $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$ can be enumerated in polynomial time (see Lemma 3.16 below).

Lemma 3.13.

$$\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)} = \{ \langle\!\langle U[y_1^{(n)}(T_1, \ldots, T_n)] \rangle\!\rangle_{\vec{v}} \mid U \in \mathbb{P}_{\Delta,k}, n \leq p, T_1, \ldots, T_n \in \mathbb{T}_\Delta, \vec{v} \in \{0,1\}^n \}.$$

Proof:
Write $\mathbb{C}'$ for the right-hand side of the equation.

($\subseteq$). Let $G = (N, \mathbb{O}, \sigma, F, P, I) \in \mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$. It suffices to show that for every $A \in N$, the yield C of every (A)-derivation environment ξ belongs to $\mathbb{C}'$. We prove this by induction on ξ.

Case 1. $\xi = \Box^A$. Then $A \in I$ and C is the identity function on $\sigma(A) = \mathbb{T}_\Delta$. So $C = \langle\!\langle y_1^{(0)} \rangle\!\rangle_{()}$ belongs to $\mathbb{C}'$.

Case 2. ξ is the result of substituting $\pi(\tau_1, \dots, \tau_{i-1}, \Box^{A_i}, \tau_{i+1}, \dots, \tau_q)$ for $\Box^{A_0}$ in an (A_0)-derivation environment ξ_0 with yield $C_0 \colon \sigma(A_0) \to \mathbb{T}_\Delta$, where for $j = 1, \dots, i-1, i+1, \dots, q$, τ_j is an A_j-derivation tree with yield U_j and $\pi = A_0(f(z_1, \dots, z_q)) \coloneq A_1(z_1), \dots, A_q(z_q)$ is a non-terminating rule ($q \geq 1$). Suppose that $\sigma(A_j) = R_{\vec{v_j}}$ with $\vec{v_j} \in \{0,1\}^{n_j}$ for $j = 0, \dots, q$. Then $f = \langle\!\langle Q \rangle\!\rangle_{\vec{u_1}, \dots, \vec{u_q}}$ for some k-copying tree pattern $Q \in \mathbb{T}_\Delta\langle Y_q\rangle[X_{n_0}]$ such that Q is uniformly copying on $R_{\vec{u_1}} \times \cdots \times R_{\vec{u_q}}$ and $\vec{u_j} \geq \vec{v_j}$ ($j = 1, \dots, q$). By Lemma 3.12, we have $U_j \in \mathbb{P}_{\Delta,k}^{(n_j)}$ for $j = 1, \dots, i-1, i+1, \dots, q$.

By induction hypothesis,

$$C_0 = \langle\!\langle U[y_1^{(n_0)}(T_1, \dots, T_{n_0})] \rangle\!\rangle_{\vec{v_0}}$$

for some $U \in \mathbb{P}_{\Delta,k}$ and $T_1, \dots, T_{n_0} \in \mathbb{T}_\Delta$. By the definition of the yield of an (A_i)-environment,

$$C(z) = C_0(f(U_1, \dots, U_{i-1}, z, U_{i+1}, \dots, U_q))$$

for every $z \in R_{\vec{v_i}}$. Then

$$\begin{aligned} C(z) &= (U[y_1^{(n_0)}(T_1, \dots, T_{n_0})])\langle Q\langle U_1, \dots, U_{i-1}, z, U_{i+1}, \dots, U_q\rangle\rangle \\ &= U[Q\langle U_1, \dots, U_{i-1}, z, U_{i+1}, \dots, U_q\rangle[T_1, \dots, T_{n_0}]] \\ &= U[Q[T_1, \dots, T_{n_0}]\langle U_1, \dots, U_{i-1}, z, U_{i+1}, \dots, U_q\rangle]. \end{aligned}$$

Since $y_i^{(n_i)}$ occurs exactly once in Q, we can write

$$Q[T_1, \dots, T_{n_0}] = V_0[y_i^{(n_i)}(V_1, \dots, V_{n_i})],$$

where V_0 is a 1-copying one-variable tree pattern and $V_1, \dots, V_{n_i}$ are trees over $\Delta \cup \{y_1^{(n_1)}, \dots, y_{i-1}^{(n_{i-1})}, y_{i+1}^{(n_{i+1})}, \dots, y_q^{(n_q)}\}$. Then

$$C = \langle\!\langle U[\overline{V_0}[y_1^{(n_i)}(\overline{V_1}, \dots, \overline{V_{n_i}})]] \rangle\!\rangle_{\vec{v_i}},$$

where for $j = 0, \dots, n_i$, $\overline{V_j} = V_j\langle U_1, \dots, U_{i-1}, U_i, U_{i+1}, \dots, U_q\rangle$ for some $U_i \in \mathbb{P}_{\Delta,k}^{(n_i)}$. (Since $y_i^{(n_i)}$ does not occur in $V_0, V_1, \dots, V_q$, it does not matter what U_i is.) Since $U_j \in \mathbb{P}_{\Delta,k}^{(n_j)}$ for $j = 1, \dots, q$, Lemma 3.10 implies $\overline{V_0} \in \mathbb{P}_{\Delta,k}$. Since $U \in \mathbb{P}_{\Delta_k}$, we have $U[\overline{V_0}[x_1]] \in \mathbb{P}_{\Delta,k}$, and it follows that $C \in \mathbb{C}'$.

($\supseteq$). Suppose $U \in \mathbb{P}_{\Delta,k}$, $n \leq p$, $T_1, \dots, T_n \in \mathbb{T}_\Delta$, $\vec{v} \in \{0,1\}^n$. We show that $\langle\!\langle U[y_1^{(n)}(T_1, \dots, T_n)] \rangle\!\rangle_{\vec{v}} \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$. Since $\langle\!\langle y_1^{(n)}(T_1, \dots, T_n) \rangle\!\rangle_{\vec{v}} \colon R_{\vec{v}} \to \mathbb{T}_\Delta$ is clearly in $\mathbb{F}_1'$, it suffices, by part (iii) of Lemma 2.2, to show by induction on $U \in \mathbb{P}_{\Delta,k}$ that $\langle\!\langle U[y_1^{(0)}] \rangle\!\rangle \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$.

Case 1. $U = x_1$. In this case, $\langle\!\langle U[y_1^{(0)}] \rangle\!\rangle$ is the identity function on $\mathbb{T}_\Delta$ and clearly belongs to $\mathbb{F}_1'$ and hence to $\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$.

Case 2. $U = U_1[U_2]$, where $U_1 \in \mathbb{P}_{\Delta,k}$ and U_2 is a k-copying one-variable tree pattern over Δ. By induction hypothesis, $\langle\!\langle U_1[y_1^{(0)}]\rangle\!\rangle \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$. Since $\langle\!\langle U_2[y_1^{(0)}]\rangle\!\rangle$ clearly belongs to $\mathbb{F}'_1$, it follows that $\langle\!\langle U_1[U_2[y_1^{(0)}]]\rangle\!\rangle$ belongs to $\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$. □

We assume that a partial function in $\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$ is represented by a tree of the form $U[y_1^{(n)}(T_1,\dots,T_n)]$ with $n \le p$, $U \in \mathbb{P}_{\Delta,k}$, and $T_1,\dots,T_n \in \mathbb{T}_\Delta$, together with $\vec{v} \in \{0,1\}^n$.

Lemma 3.14. Assumption 5 holds of $\mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$.

Proof:
Given Lemma 3.12, it suffices to show that the three items of Assumption 5 hold for $S \in \bigcup_{n=0}^{p} \mathbb{P}_{\Delta,k}^{(n)}$ and $C \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$. The first two items are trivial and the third item easily follows from Lemma 3.1. □

Lemma 3.15. Given $T \in \mathbb{T}_\Delta$, the set $\{\, Q \in \mathbb{P}_{\Delta,k}^{(n)} \mid Q \text{ matches } T \,\}$ can be enumerated in polynomial time in the size of T.

Proof:
In this proof, we assume the standard way of referring to nodes of a tree by strings of positive integers, as explained in the appendix (Section A). For $u, v \in (\mathbb{N} - \{0\})^*$, we write $u \le v$ to mean u is a prefix of v.

In order to enumerate n-variable tree patterns $Q \in \mathbb{P}_{\Delta,k}^{(n)}$ that match T, it suffices to enumerate n-tuples of one-variable tree patterns $Q_1,\dots,Q_n \in \mathbb{P}_{\Delta,k}$ that satisfy the following conditions:

- Q_i matches T for $i = 1,\dots,n$.
- If x_1 labels node u in Q_i and x_1 labels node v in Q_j for some i,j such that $1 \le i < j \le n$, then $u \not\le v$ and $u \not\ge v$.

For, if $Q \in \mathbb{P}_{\Delta,k}^{(n)}$ is such that $Q[T_1,\dots,T_n] = T$, then $Q_i = Q[T_1,\dots,T_{i-1},x_1,T_{i+1},\dots,T_n]$ clearly satisfy these conditions. Conversely, let $Q_1,\dots,Q_n \in \mathbb{P}_{\Delta,k}$ satisfy these conditions. For each $i = 1,\dots,n$, let T_i be the tree such that $Q_i[T_i] = T$. Define a tree Q whose set of nodes is the intersection of the sets of nodes of $Q_1,\dots,Q_l$ such that the label of a node u of Q is determined by the following rule:

- If x_1 labels u in Q_i, then the label of u in Q is x_i.
- If there is no i such that x_1 labels u in Q_i, then the label of u in Q is the same as the label of u in T.

Then it is easy to see that $Q_i = Q[T_1,\dots,T_{i-1},x_1,T_{i+1},\dots,T_n]$ and so $Q \in \mathbb{P}_{\Delta,k}^{(n)}$. □

Lemma 3.16. Assumption 6 holds of $\mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$.

Proof:
Firstly, we can see that $\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}|_T$ can be enumerated in polynomial time in the size of $T \in \mathbb{T}_\Delta$ by noticing

$$\begin{aligned}&\mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}|_T = \\ &\qquad \{\, \langle\!\langle U[y_1^{(n)}(T_1,\dots,T_n)]\rangle\!\rangle_{\vec{v}} \mid U \in \mathbb{P}_k, T_1,\dots,T_n \in \mathbb{T}_\Delta, \vec{v} \in \{0,1\}^n, \text{and} \\ &\qquad\qquad T = U[V[T_1,\dots,T_n]] \text{ for some 1-copying } V \in \mathbb{T}_\Delta[X_n] \,\}.\end{aligned}$$

This is because if $T = U[V'[T_1,\dots,T_n]]$ for some $V' \in \mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}$, then we can obtain a 1-copying $V \in \mathbb{T}_\Delta[X_n]$ such that $T = U[V[T_1,\dots,T_n]]$ by replacing all but the first occurrence of x_i in V' by T_i for each $i = 1,\dots,n$, and it is clear that any 1-copying $V \in \mathbb{T}_\Delta[X_n]$ belongs to $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}$.

Similarly, we can see

$$\begin{aligned}\mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}|_T = \bigcup_{q=0}^{r} \{\, f \in \mathbb{F}'_q \mid\ & C(f(S_1,\dots,S_q)) = T, C \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}|_T, \\ & S_i \in \mathbb{T}_\Delta[X_{n_i}] \text{ and } S_i \text{ is 1-copying for } i = 1,\dots,q \,\},\end{aligned}$$

which shows that $\mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}|_T$ is enumerable in polynomial time.

It remains to show that $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$ is enumerable in polynomial time. Write $\mathbb{S}'$ for the right-hand side of the inclusion in Lemma 3.12. In order to enumerate $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$, we can first enumerate

$$\mathbb{S}'|_T = \{\, S \in \mathbb{S}' \mid C(S) = T \text{ for some } C \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}|_T \,\},$$

and check for each of its elements whether it is in $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$ by dynamic programming, using the functions in $\mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}|_T$. Thus, any 0-ary function in $\mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}|_T$ that is in $\mathbb{S}'|_T$ must be in $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$. Any element S of $\mathbb{S}'|_T$ for which there are substructures $S_1,\dots,S_q \in \mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$ and a q-ary function $f \in \mathbb{F}_{\text{UC-IOCFTG}(k,p,r)}|_T$ such that $f(S_1,\dots,S_q)$ is defined and equals S must also be in $\mathbb{S}_{\text{UC-IOCFTG}(k,p,r)}|_T$. The fact that $\mathbb{S}'|_T$ can be enumerated in polynomial time follows from Lemma 3.15, since if $C(S) = T$ for some $C \in \mathbb{C}_{\text{UC-IOCFTG}(k,p,r)}$, S must match some subtree of T. □

Assumptions 1, 2, 3, and 4 are all easy to check, so we have

Theorem 3.17. Algorithms 1 and 2, when applied to $\mathbb{G} = \mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$, successfully learn all grammars in $\mathbb{G}_{\text{UC-IOCFTG}(k,p,r)}$ with the m-FKP and the m-FCP, respectively, while satisfying favorable properties in terms of efficiency.

4. Conclusion

In previous works on distributional learning of nonlinear grammars [14, 2], the question of polynomial enumerability of the sets $\mathbb{S}|_T$ and $\mathbb{C}|_T$ of possible substructures/contexts contained in a given element T of input data has not been given enough scrutiny. These papers gave the impression that the nonlinearity of the grammatical operations (functions) automatically implies that at least one of these sets becomes too big to be enumerated in polynomial time. In this paper, we have shown that that is not the case, and presented two classes of nonlinear tree grammars for which distributional learning is possible in its "pristine" form, succeeding on all grammars with the m-FKP/m-FCP. The first class—the class of parallel regular tree grammars (with certain parameters fixed)—has a natural definition similar to the classes targeted by previous results on distributional learning, but its expressive power is rather limited. It is easy to see that the class of tree languages generated by parallel regular tree grammars consists of all homomorphic images of regular tree languages. The second class—the class of uniformly copying IO context-free tree grammars—extends simple context-free tree grammars and is much more expressive, but "uniform copying" is a qualitative restriction on IO context-free tree grammars with a rather convoluted definition whose sole motivation is the possibility of successful distributional learning. It remains to be seen whether any independently motivated natural constraint on rules of nonlinear grammars similarly leads to polynomial enumerability of substructures and contexts and hence to an interesting class amenable to distributional learning.

References

[1] Clark, A.: Learning context free grammars with the syntactic concept lattice, in: Sempere and García [11], 38–51.

[2] Clark, A., Yoshinaka, R.: Distributional learning of parallel multiple context-free grammars, *Machine Learning*, **96**(1–2), 2014, 5–31.

[3] Engelfriet, J., Schmidt, E. M.: IO and OI, part I, *The Journal of Computer and System Sciences*, **15**, 1977, 328–353.

[4] Fischer, M. J.: *Grammars with Macro-Like Productions*, Ph.D. Thesis, Harvard University, 1968.

[5] Kanazawa, M., Yoshinaka, R.: Distributional learning and context/substructure enumerability in nonlinear tree grammars, *Proceedings of Formal Grammar 2015* (A. Foret, G. Morrill, R. Muskens, R. Osswald, Eds.), to appear.

[6] Kasprzik, A., Yoshinaka, R.: Distributional learning of simple context-free tree grammars, *Algorithmic Learning Theory* (J. Kivinen, C. Szepesvári, E. Ukkonen, T. Zeugmann, Eds.), Lecture Notes in Computer Science, Springer, 2011.

[7] Leiß, H.: Learning context free grammars with the finite context property: A correction of A. Clark's algorithm, *Formal Grammar* (G. Morrill, R. Muskens, R. Osswald, F. Richter, Eds.), Lecture Notes in Computer Science, Springer, 2014.

[8] Osherson, D. N., Stob, M., Weinstein, S.: *Systems That Learn: An Introduction to Learning Theory for Cognitive and Computer Scientists*, The MIT Press, Cambridge, MA, 1986.

[9] Pollard, C. J.: *Generalized Phrase Structure Grammars, Head Grammars, and Natural Language*, Ph.D. Thesis, Stanford University, 1984.

[10] Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars, *Theoretical Computer Science*, **88**, 1991, 191–229.

[11] Sempere, J. M., García, P., Eds.: *Grammatical Inference: Theoretical Results and Applications, 10th International Colloquium, ICGI 2010*, Springer, 2010.

[12] Yoshinaka, R.: Polynomial-time identification of multiple context-free languages from positive data and membership queries, in: Sempere and García [11], 230–244.

[13] Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices, *Developments in Language Theory* (G. Mauri, A. Leporati, Eds.), Lecture Notes in Computer Science, Springer, 2011, ISBN 978-3-642-22320-4.

[14] Yoshinaka, R.: An attempt towards learning semantics: Distributional learning of IO context-free tree grammars, *Proceedings of the 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, 2012.

[15] Yoshinaka, R., Kanazawa, M.: Distributional learning of abstract categorial grammars, *Logical Aspects of Computational Linguistics* (S. Pogodalla, J.-P. Prost, Eds.), Lecture Notes in Computer Science, Springer, 2011, ISBN 978-3-642-22220-7.

A. Enumerating One-Variable Tree Patterns That Match a Given Tree

In this appendix, we mainly think of trees in terms of their graphical representation. Thus, we use terms like *node, parent, child, root, leaf* with their familiar meanings. Recall that the *size* $|T|$ of a tree T is the number of nodes of T. The *height* of a node in T is 0 if it is a leaf and 1 plus the maximal height of its children otherwise.

We adopt a standard way of referring to a node in a tree T by a string over $\mathbb{N} - \{0\}$. Thus, ε (the empty string) refers to the root of T, and if $u \in (\mathbb{N} - \{0\})^*$ refers to a node of T, then the ith child of u (if it exists) is referred to by ui. When $u = i_1 \dots i_n \in (\mathbb{N} - \{0\})^n$ for $n \geq 1$, we write u^- for $i_1 \dots i_{n-1}$ and $\mathrm{last}(u)$ for i_n. If u is a node of T, we write T/u for the subtree of T rooted at u. Note that when uv is a node of T, we always have $T/uv = (T/u)/v$. A *path* (to a node u) is a sequence of nodes of the form $\varepsilon, i_1, i_1 i_2, \dots, i_1 i_2 \dots i_n = u$. Note that a path can contain at most one node of a given height h.

We write a one-variable tree pattern $P \in \mathbb{T}_\Delta[X_1]$ as $P[x_1]$ in order to clearly distinguish one-variable tree patterns from trees in $\mathbb{T}_\Delta$. Recall that a one-variable tree pattern $P[x_1]$ *matches* a tree T if there exists a tree T' such that $P[T'] = T$.

The following example shows that there are in general exponentially many one-variable tree patterns that match a given tree.

Example A.1. Consider the perfect binary tree T_h over $\{f, a\}$ of height h, defined by

$$T_0 = a,$$
$$T_{h+1} = f(T_h, T_h).$$

Note that T_h has 2^h leaves and $n = 2^{h+1} - 1$ nodes. We can replace any nonempty subset of the leaves of T_h by x_1 to obtain a one-variable tree pattern $P[x_1]$ such that $P[a] = T_h$, so there are at least $2^{2^h} - 1 = 2^{\frac{n+1}{2}} - 1$ one-variable tree patterns that match T_h.

Within a polynomial time bound, we can only enumerate a subset of the one-variable tree patterns that match a given tree. As we have noted, it is trivial to enumerate the subset consisting of *k-copying* one-variable tree patterns.

We consider a superclass of the class of k-copying one-variable tree patterns, namely, the class $\mathbb{P}_k$ of one-variable tree patterns that are compositions of k-copying one-variable tree patterns. We show that for every tree T of size n, there are no more than n^{k+1} one-variable tree patterns in $\mathbb{P}_k$ that match T.

It is easy to see that if $P[x_1] \in \mathbb{P}_k$, the prime factors of the number of occurrences of x_1 in $P[x_1]$ are all less than or equal to k. In particular, if $P[x_1] \in \mathbb{P}_2$, the number of occurrences of x_1 in $P[x_1]$ is a power of 2. This fact alone, however, does not imply that the number of one-variable tree patterns in $\mathbb{P}_k$ that match a given tree is polynomial in the size of the tree.

Example A.1. (continued) The tree T_{h+1} has $n = 2^{h+2} - 1$ nodes and $2^{h+1} = 2 \cdot 2^h$ leaves. Since any number of the first 2^h leaves may be included in a subset of the leaves of size 2^h, the number of subsets of the leaves of size 2^h clearly exceeds $2^{2^h} = 2^{\frac{n+1}{4}}$. So there are exponentially many one-variable tree patterns $P[x_1]$ with 2^h occurrences of x_1 such that $P[a] = T_{h+1}$. Of course, many of these one-variable tree patterns are not elements of $\mathbb{P}_2$. For example, it is easy to see that $f(f(f(x_1, x_1), f(x_1, a)), f(f(x_1, a), f(a, a))) \not\in \mathbb{P}_2$. Out of the $\binom{8}{4} = 70$ one-variable tree patterns $P[x_1]$ with 4 occurrences of x_1 such that $P[a] = T_3$, only 10 of them are in $\mathbb{P}_2$:

$$\begin{gathered}
f(f(f(x_1, x_1), f(x_1, x_1)), f(f(a, a), f(a, a))) \\
f(f(f(x_1, x_1), f(a, a)), f(f(x_1, x_1), f(a, a))) \\
f(f(f(x_1, x_1), f(a, a)), f(f(a, a), f(x_1, x_1))) \\
f(f(f(x_1, a), f(x_1, a)), f(f(x_1, a), f(x_1, a))) \\
f(f(f(x_1, a), f(a, x_1)), f(f(x_1, a), f(a, x_1))) \\
f(f(f(a, x_1), f(x_1, a)), f(f(a, x_1), f(x_1, a))) \\
f(f(f(a, x_1), f(a, x_1)), f(f(a, x_1), f(a, x_1))) \\
f(f(f(a, a), f(x_1, x_1)), f(f(x_1, x_1), f(a, a))) \\
f(f(f(a, a), f(x_1, x_1)), f(f(a, a), f(x_1, x_1))) \\
f(f(f(a, a), f(a, a)), f(f(x_1, x_1), f(x_1, x_1)))
\end{gathered}$$

For example, the first of these is the composition of $f(f(x_1, x_1), f(f(a, a), f(a, a)))$ and $f(x_1, x_1)$. It is not difficult to see that there are exactly 47 one-variable tree patterns $P[x_1]$ in $\mathbb{P}_2$ such that $P[a] = T_3$, and a total of 62 one-variable tree patterns in $\mathbb{P}_2$ that match T_3.

The next example shows that the number of sequences of k-copying one-variable tree patterns whose composition matches a tree of size n is in general not bounded by any polynomial function of n, even when those involving 1-copying one-variable tree patterns are excluded.

Example A.2. For each $m \in \mathbb{N}$, define a tree U_m by

$$\begin{aligned}
U_0 &= a \\
U_{m+1} &= f(g^m(U_m)), g^m(U_m)),
\end{aligned}$$

where $g^i(T)$ abbreviates

$$\underbrace{g(\dots(g(}_{i \text{ times}} T \underbrace{)\dots)}_{i \text{ times}}.$$

We have

$$\begin{aligned}
|U_0| &= 1, \\
|U_{m+1}| &= 2|U_m| + 2m + 1,
\end{aligned}$$

and it easily follows that $|U_m| \le 3^{m+1}$. For example,

$$U_3 = f(g(g(f(g(f(a, a)), g(f(a, a))))), g(g(f(g(f(a, a)), g(f(a, a)))))).$$

and $|U_3| = 23 < 3^4 = 81$.

Now let $m \geq 1$ and for each $i = 0, \dots, m-1$, pick an $l_i \in \{0, \dots, i\}$. Define a 2-copying one-variable tree pattern $P_i[x_1]$ for each $i = 0, \dots, m$ by

$$\begin{aligned} P_0[x_1] &= x_1, \\ P_i[x_1] &= g^{l_i}(f(g^{i-1-l_{i-1}}(x_1), g^{i-1-l_{i-1}}(x_1))) \quad \text{for } i = 1, \dots, m-1, \\ P_m[x_1] &= f(g^{m-1-l_{m-1}}(x_1), g^{m-1-l_{m-1}}(x_1)). \end{aligned}$$

Then

$$P_m[\dots[P_0[x_1]]\dots] = P_m[\dots[P_1[x_1]]\dots]$$

matches U_m. There are $m!$ ways of picking $l_0, \dots, l_{m-1}$, which result in $m!$ distinct sequences of 2-copying one-variable tree patterns. Note that $m!$ is not $O((3^{m+1})^c)$ for any constant c. However, all these sequences compose to the same one-variable tree pattern in $\mathbb{P}_2$, namely, the result of replacing all occurrences of a in U_m by x_1.

Definition A.3. A one-variable tree pattern $P[x_1]$ is said to be *right-reduced* if $P[x_1]$ has leaves u, v labeled by x_1 such that $u, v \in (\mathbb{N} - \{0\})^+$ and either

1. $\text{last}(u) \neq \text{last}(v)$, or
2. $P[x_1]/u^- \neq P[x_1]/v^-$.

Lemma A.4. Let $P[x_1]$ be a one-variable tree pattern such that $P[x_1] \neq x_1$. The following are equivalent:

(i) $P[x_1]$ is right-reduced.

(ii) If $P[x_1] = P_1[P_2[x_1]]$ and $P_2[x_1]$ is 1-copying, then $P_2[x_1] = x_1$.

Proof:
The direction (i) $\Rightarrow$ (ii) is easy. To see the converse direction, assume that $P[x_1]$ is not right-reduced. Since $P[x_1] \neq x_1$, every leaf of $P[x_1]$ is in $(\mathbb{N} - \{0\})^+$. Since $P[x_1]$ is not right-reduced, every pair of leaves u, v of $P[x_1]$ labeled by x_1 must satisfy $\text{last}(u) = \text{last}(v)$ and $P[x_1]/u^- = P[x_1]/v^-$. It easily follows that v cannot be a descendant of u^- in $P[x_1]$ and hence $P[x_1]/u^-$ is 1-copying. If $P'[x_1]$ is the result of replacing all occurrences of $P[x_1]/u^-$ in $P[x_1]$ by x_1, then $P[x_1] = P'[P[x_1]/u^-]$. Since $P[x_1]/u^-$ has at least two nodes, condition (ii) does not hold. □

Definition A.5. A sequence $P_1[x_1], \dots, P_{m+1}[x_1]$ ($m \geq 0$) of k-copying one-variable tree patterns is said to be *standard* if $P_{m+1}[x_1]$ is 1-copying and for each $i = 1, \dots, m$, $P_i[x_1]$ is right-reduced.

Example A.2. (continued) The sequence $P_m[x_1], \dots, P_0[x_1]$ is standard if and only if $l_i = i$ for all $i = 0, \dots, m-1$. In this case, we have

$$\begin{aligned} P_0[x_1] &= x_1, \\ P_i[x_1] &= g^i(f(x_1, x_1)) \quad \text{for } i = 1, \dots, m-1, \\ P_m[x_1] &= f(x_1, x_1). \end{aligned}$$

The following lemma implies that the number of standard sequences of k-copying one-variable tree patterns whose composition matches a tree T is an upper bound on the number of one-variable tree patterns in $\mathbb{P}_k$ that match T.

Lemma A.6. For every sequence of k-copying one-variable tree patterns $P_1[x_1], \dots, P_m[x_1]$ $(m \geq 1)$, there exists a standard sequence of k-copying one-variable tree patterns $P'_1[x_1], \dots, P'_{m'}[x_1]$ such that

$$P_1[\dots[P_m[x_1]]\dots] = P'_1[\dots[P'_{m'}[x_1]]\dots].$$

Proof:
Let $P_1[x_1], \dots, P_m[x_1]$ be k-copying one-variable tree patterns. Without loss of generality, we may assume that $P_m[x_1]$ is the only 1-copying one-variable tree pattern among $P_1[x_1], \dots, P_m[x_1]$. For, if $P_m[x_1]$ is not 1-copying, we can just add x_1 at the end of the sequence, and if $P_i[x_1]$ is 1-copying for some $i < m$, we can replace the subsequence $P_i[x_1], P_{i+1}[x_1]$ by $P_i[P_{i+1}[x_1]]$, which is still k-copying. The following algorithm computes the required standard sequence.

```
1. Let $P'_i[x_1] := P_i[x_1]$ for $i = 1, \dots, m$.
2. Let $j := m - 1$.
3. While $j > 0$, repeat the following:
   (a) If $P'_j[x_1]$ is right-reduced, let $j := j - 1$.
   (b) Otherwise, let $Q[x_1] := P'_j[x_1]/u^-$, where $u$ is a leaf of $P'_j[x_1]$ labeled by $x_1$. Let $P''_j[x_1]$ be
       the result of replacing all occurrences of $Q[x_1]$ in $P'_j[x_1]$ by $x_1$. Let $P'_{j+1}[x_1] := Q[P'_{j+1}[x_1]]$
       and $P'_j[x_1] := P''_j[x_1]$.
```

At all stages of the execution of the algorithm, $P'_{j+1}[x_1], \dots, P'_m[x_1]$ is a standard sequence of k-copying one-variable tree patterns. In line (3b), $P'_j[x_1]/u^-$ is 1-copying and $P'_j[x_1] = P''_j[Q[x_1]]$ by Lemma A.4. This implies that $Q[P'_{j+1}[x_1]]$ and $P''_j[x_1]$ are k-copying, $Q[P'_{j+1}[x_1]]$ is right-reduced, and $P'_1[\dots[P'_m[x_1]]\dots]$ remains the same. Since every iteration of the while loop decreases j or the size of $P'_j[x_1]$, the algorithm terminates. □

Definition A.7. A pair of nodes (u, v) of a tree T is said to be *productive* if the following conditions hold:

1. $u, v \in (\mathbb{N} - \{0\})^+$,
2. $T/u = T/v$, and
3. either
   (a) $\mathrm{last}(u) \neq \mathrm{last}(v)$, or
   (b) $T/u^- \neq T/v^-$.

Lemma A.8. Let u, v be distinct nodes of a tree T such that $T/u = T/v$. Then there exist a productive pair of nodes (u', v') and a string $z \in (\mathbb{N} - \{0\})^*$ such that $u = u'z$ and $v = v'z$.

Proof:
Since $u \neq v$, either (u, v) is itself productive or $\mathrm{last}(u) = \mathrm{last}(v)$, $u^- \neq v^-$, and $T/u^- = T/v^-$. The lemma is easily shown by induction. □

Lemma A.9. Let T be a tree with n nodes, and suppose that for each $l = 1, \dots, p$, $u_l, v_l \in (\mathbb{N} - \{0\})^+$ and

$$(u_1 \dots u_l, v_1 \dots v_l)$$

is a productive pair of nodes of T. Then $p < \log_2 n$.

Proof:
Let $u_0 = v_0 = \varepsilon$. Then for each $l = 1, \dots, p$, $T/(u_1 \dots u_{l-1}) = T/(v_1 \dots v_{l-1})$. Since $(u_1 \dots u_l, v_1 \dots v_l)$ is a productive pair, either $\mathrm{last}(u_1 \dots u_l) \neq \mathrm{last}(v_1 \dots v_l)$ or $T/(u_1 \dots u_l)^- \neq T/(v_1 \dots v_l)^-$. In the former case, since $\mathrm{last}(u_1 \dots u_l) = \mathrm{last}(u_l)$ and $\mathrm{last}(v_1 \dots v_l) = \mathrm{last}(v_l)$, we have $\mathrm{last}(u_l) \neq \mathrm{last}(v_l)$. In the latter case, since $(u_1 \dots u_l)^- = u_1 \dots u_{l-1} u_l^-$, $(v_1 \dots v_l)^- = v_1 \dots v_{l-1} v_l^-$, and $T/u_1 \dots u_{l-1} = T/v_1 \dots v_{l-1}$, we must have $u_l^- \neq v_l^-$. In either case, we have $u_l \neq v_l$ and $u_1 \dots u_{l-1} u_l$ and $u_1 \dots u_{l-1} v_l$ are two distinct descendants of $u_1 \dots u_{l-1}$ such that $T/u_1 \dots u_{l-1} u_l = T/v_1 \dots u_{l-1} v_l$. So we have

$$|T/u_1 \dots u_{l-1}| > 2|T/u_1 \dots u_l|$$

for $l = 1, \dots, p$, which implies

$$|T| = |T/u_0| > 2^p |T/u_1 \dots u_p| \geq 2^p.$$

Therefore, $p < \log_2 |T|$. □

Lemma A.10. Let $P_1[x_1], \dots, P_{m+1}[x_1]$ be a standard sequence of k-copying one-variable tree patterns. For each $l = 1, \dots, m$, there are distinct leaves u_l, v_l of $P_l[x_1]$ labeled by x_1 satisfying the following condition:

- if $P_1[\dots[P_{m+1}[x_1]]\dots]$ matches a tree T, then for each $l = 1, \dots, m$,

$$(u_1 \dots u_l, v_1 \dots v_l)$$

is a productive pair of nodes of T.

Proof:
Since $P_1[x_1], \dots, P_{m+1}[x_1]$ is a standard sequence, for each $l = 1, \dots, m$, there exist one or more pairs of nodes (u, v) witnessing the fact that $P_l[x_1]$ is right-reduced (see Definition A.3). We pick one particular such pair (u_l, v_l) according to the following rule. (Not every witnessing pair will do.)

Case 1. $P_l[x_1]$ has two leaves u, v labeled by x_1 such that $\mathrm{last}(u) \neq \mathrm{last}(v)$. Then we let $u_l = u$ and $v_l = v$.

Case 2. There exists an i such that all leaves u of $P_l[x_1]$ labeled by x_1 are of the form $u^- i$.

Case 2.1. There exist a leaf u of $P_l[x_1]$ labeled by x_1, $j \neq i$, and $w \in (\mathbb{N} - \{0\})^*$ such that $u^- jwi$ is a leaf of $P_l[x_1]$ labeled by x_1. In this case, let $u_l = u$ and $v_l = u^- jwi$. Since v_l^- is a descendant of u_l^- in $P_l[x_1]$, we have $P_l[x_1]/u_l^- \neq P_l[x_1]/v_l^-$.

Case 2.2. For all leaves u of $P_l[x_1]$ labeled by x_1, u is the only descendant of u^- in $P_l[x_1]$ labeled by x_1. In this case, take any pair of leaves u_l, v_l of $P_l[x_1]$ labeled by x_1 such that $P_l[x_1]/u_l^- \neq P_l[x_1]/v_l^-$, which must exist.

Now suppose $T = P_1[\dots[P_{m+1}[T']]\dots]$. We show that $(u_1 \dots u_l, v_1 \dots v_l)$ is a productive pair of nodes of T for $l = 1, \dots, m$.

Clearly, $u_1 \dots u_l$ and $v_1 \dots v_l$ belong to $(\mathbb{N} - \{0\})^+$, so the first condition of Definition A.7 is satisfied. Also, it is clear that $T/u_1 \dots u_l = T/v_1 \dots v_l = P_{l+1}[\dots[P_{m+1}[T']]\dots]$, which takes care of the second condition.

It remains to show the third condition of Definition A.7. We reason according to the case distinction in the definition of u_l, v_l.

If Case 1 applied to $P_l[x_1]$, then $\mathrm{last}(u_l) \neq \mathrm{last}(v_l)$, so $\mathrm{last}(u_1 \dots u_l) \neq \mathrm{last}(v_1 \dots v_l)$, satisfying the clause (3a) of Definition A.7.

If Case 2.1 applied to $P_l[x_1]$, then $u_1 \dots u_{l-1} v_l^-$ is a descendant of $u_1 \dots u_{l-1} u_l^-$ in T, so $T/u_1 \dots u_{l-1} u_l^- \neq T/u_1 \dots u_{l-1} v_l^-$. Since $T/(u_1 \dots u_l)^- = T/u_1 \dots u_{l-1} u_l^-$ and $T/(v_1 \dots v_l)^- = T/v_1 \dots v_{l-1} v_l^- = P_l[\dots[P_{m+1}[T']]\dots]/v_l^- = T/u_1 \dots u_{l-1} v_l^-$, we have $T/(u_1 \dots u_l)^- \neq T/(v_1 \dots v_l)^-$, satisfying the clause (3b) of Definition A.7.

Suppose Case 2.2 applied to $P_l[x_1]$. Since $P_l[x_1]/u_l^- \neq P_l[x_1]/v_l^-$, one of the following three conditions must obtain:

(i) u_l^- and v_l^- have different labels in $P_l[x_1]$.

(ii) u_l^- and v_l^- have different numbers of children in $P_l[x_1]$.

(iii) There exist a $j \neq i$ such that $u_l^- j$ and $v_l^- j$ are both nodes of $P_l[x_1]$ and $P_l[x_1]/u_l^- j \neq P_l[x_1]/v_l^- j$.

If (i) or (ii) obtains, it is clear that $P_l[P_{l+1}[\dots[P_m[T']]\dots]]/u_l^-$ and $P_l[P_{l+1}[\dots[P_m[T']]\dots]]/v_l^-$ differ in the same way. In the case of (iii), since u_l is the only descendant of u_l^- in $P_l[x_1]$ that is labeled by x_1 and v_l is the only descendant of v_l^- in $P_l[x_1]$ that is labeled by x_1, $P_l[x_1]/u_l^- j = P_l[P_{l+1}[\dots[P_m[T']]\dots]]/u_l^- j$ and $P_l[x_1]/v_l^- j = P_l[P_{l+1}[\dots[P_m[T']]\dots]]/v_l^- j$, which again implies $P_l[P_{l+1}[\dots[P_m[T']]\dots]]/u_l^- \neq P_l[P_{l+1}[\dots[P_m[T']]\dots]]/v_l^-$. So $T/(u_1 \dots u_l)^- = T/u_1 \dots u_{l-1} u_l^- \neq T/v_1 \dots v_{l-1} v_l^- = T/(v_1 \dots v_l)^-$ holds in all three cases (i)–(iii) and the clause (3b) is satisfied. □

If $w_1, \dots, w_k$ are nodes of T such that $T/w_1 = \dots = T/w_k$, then we let $T_{\{w_1,\dots,w_k\}}[x_1]$ denote the unique k-copying one-variable tree pattern $P[x_1]$ such that $w_1, \dots, w_k$ are the leaves of $P[x_1]$ labeled by x_1 and $P[T/w_1] = T$.

Lemma A.11. Suppose that $w_1, \dots, w_k$ are nodes of T such that $T/w_1 = \dots = T/w_k$. If (w_1, w_2) is a productive pair of nodes of T, then $T_{\{w_1,\dots,w_k\}}[x_1]$ is a right-reduced k-copying one-variable tree pattern.

Proof:
Assume that (w_1, w_2) is a productive pair of nodes of T. Then either $\mathrm{last}(w_1) \neq \mathrm{last}(w_2)$ or $T/w_1^- \neq T/w_2^-$. In the former case, $T_{\{w_1,\dots,w_k\}}[x_1]$ is right-reduced. Suppose $T/w_1^- = T/w_2^-$, and let

$$Q_1[x_1] = T_{\{w_1,\dots,w_k\}}[x_1]/w_1^-,$$
$$Q_2[x_1] = T_{\{w_1,\dots,w_k\}}[x_1]/w_2^-.$$

Then $T/w_1^- = Q_1[T/w_1]$ and $T/w_2^- = Q_2[T/w_1]$. Since $T/w_1^- \neq T/w_2^-$, we have $Q_1[x_1] \neq Q_2[x_1]$ and so $T_{\{w_1,\dots,w_k\}}$ is right-reduced. □

Theorem 3.2. Let T be a tree with n nodes. There are no more than n^{k+1} one-variable tree patterns in $\mathbb{P}_k$ that match T, and they can be enumerated in polynomial time.

Proof:
We first describe informally a nondeterministic procedure for computing a standard sequence $P_1[x_1], \dots, P_{m+1}[x_1]$ of k-copying one-variable tree patterns whose composition matches T.

Let u, v be two nodes of T such that

$$T/u = T/v = T'. \tag{6}$$

Let

$$(\hat{u}_1, \hat{v}_1), \dots, (\hat{v}_p, \hat{v}_p)$$

list in the order of decreasing height the productive pairs of nodes of T whose first component lies on the path to u and whose second component lies on the path to v.

Case 1. If $u = v$, then $p = 0$ and the node u determines a 1-copying one-variable tree pattern

$$P_1[x_1] = T_{\{u\}}[x_1] \tag{7}$$

such that

$$P_1[T'] = T.$$

Case 2. Otherwise, by Lemma A.8, $p \geq 1$ and

$$u = \hat{v}_p z, \quad v = \hat{v}_p z$$

for some z. By Lemma A.9, we have $p < \log_2 n$. Pick $m \leq p$ of these pairs, including $(\hat{v}_p, \hat{v}_p)$, which must be of the form

$$(u_1, v_1), (u_1 u_2, v_1 v_2), \dots, (u_1 \dots u_m, v_1 \dots v_m).$$

We have

$$u = u_1 \dots u_m z, \quad v = v_1 \dots v_m z.$$

Let h_l be the height of $u_1 \dots u_l$ in T (which must the same as the height of $v_1 \dots v_l$ in T since $T/u_1 \dots u_l = T/v_1 \dots v_l$). Now pick $k' \leq k - 2$ distinct nodes w_i' ($i = 1, \dots, k'$) of T that satisfy the following conditions:

$$w_i = w_{i,1} \dots w_{i,m} z, \tag{8}$$

$$T/w_{i,1} \dots w_{i,l} = T/u_1 \dots u_l \quad \text{for } l = 1, \dots, m, \tag{9}$$

where $w_{i,1} \dots w_{i,l}$ is the unique node of height h_l on the path to w_i. (Note that this implies $T/w_1 = \dots = T/w_{k'} = T'$.) By (9), for each $l = 1, \dots, m$,

$$w_{1,l}, \quad \dots, \quad w_{k',l}, \quad u_l, \quad v_l$$

are (not necessarily pairwise distinct) nodes of $T/u_1 \dots u_{l-1}$ and we have

$$(T/u_1 \dots u_{l-1})/w_{1,l} = \dots = (T/u_1 \dots u_{l-1})/w_{k',l} = (T/u_1 \dots u_{l-1})/u_l = (T/u_1 \dots u_{l-1})/v_l.$$

Let

$$P_l[x_1] = (T/u_1 \dots u_{l-1})_{\{w_{1,l},\dots,w_{k',l},u_l,v_l\}}[x_1], \quad \text{for } l = 1, \dots, m, \tag{10}$$

$$P_{m+1} = (T/u_1 \dots u_m)_{\{z\}}[x_1]. \tag{11}$$

By Lemma A.11, $P_1[x_1], \dots, P_{m+1}[x_1]$ is a standard sequence of k-copying one-variable tree patterns. We have

$$P_l[T/u_1 \dots u_l] = T/u_1 \dots u_{l-1}$$

for $l = 1, \dots, m$, and

$$P_{m+1}[T'] = T/u_1 \dots u_m.$$

This implies

$$P_1[\dots[P_{m+1}[T']]\dots] = T.$$

This ends the description of our nondeterministic procedure. We now show that every standard sequence of k-copying one-variable tree patterns $P_1[x_1], \dots, P_{m+1}[x_1]$ such that $P_1[\dots[P_{m+1}[x_1]]\dots]$ matches T is obtained by the above procedure with a suitable choice of $u, v, (u_1, v_1), \dots, (u_m, v_m), w_1, \dots, w_{k'}$.

Suppose that $P_1[x_1], \dots, P_{m+1}[x_1]$ is a standard sequence of k-copying one-variable tree patterns such that x_1 occurs in $P_{m+1}[x_1]$ and

$$P_1[\dots[P_{m+1}[T']]\dots] = T.$$

Let z be the only leaf of $P_{m+1}[x_1]$ labeled by x_1. (Recall that $P_{m+1}[x_1]$ is 1-copying.) By Lemma A.10, for each $l = 1, \dots, m$, there are leaves u_l, v_l of $P_l[x_1]$ labeled by x_1 such that $(u_1 \dots u_l, v_1 \dots v_l)$ is a productive pair of nodes of T. For each $l = 1, \dots, m$, let $z_{l,1}, \dots, z_{l,k_l}$ be the leaves of $P_l[x_1]$ other than u_l, v_l that are labeled by x_1. We must have $k_l \le k - 2$ since $P_l[x_1]$ is k-copying.

Case 1. $m = 0$. Then $P_1[\dots[P_{m+1}[x_1]]\dots] = P_1[x_1]$. Let $u = v = z$. We have

$$T/u = T/v = T',$$
$$P_1[x_1] = T_{\{u\}}[x_1].$$

So (6) holds and Case 1 of the above procedure applies, giving $P_1[x_1]$ by (7).

Case 2. $m \ge 1$. Let $k' = \max\{\, k_l \mid 1 \le l \le m \,\}$ and for each $l = 1, \dots, m$ and for each $i = 1, \dots, k'$, define

$$w_{i,l} = \begin{cases} z_{l,i} & \text{if } i \le l_k, \\ u_l & \text{otherwise.} \end{cases}$$

Let

$$w_1 = w_{1,1} \dots w_{1,m} z,$$
$$\vdots$$
$$w_{k'} = w_{k',1} \dots w_{k',m} z,$$
$$u = u_1 \dots u_m z,$$
$$v = v_1 \dots v_m z.$$

Then $w_1, \dots, w_{k'}, u, v$ are $k' + 2 \leq k$ distinct nodes of T, and we have

$$T/w_{1,1} \dots w_{1,l} = \dots = T/w_{k',1} \dots w_{k',l} = T/u_1 \dots u_l = T/v_1 \dots v_l,$$
$$P_l[x_1] = (T/u_1 \dots u_{l-1})_{\{w_{1,l}, \dots, w_{k',l}, u_l, v_l\}}[x_1]$$

for each $l = 1, \dots, m$, and

$$T/w_1 = \dots = T/w_{k'} = T/u = T/v = T',$$
$$P_{m+1}[x_1] = (T/u_1 \dots u_m)_{\{z\}}[x_1].$$

Hence $u, v, (u_1, v_1), \dots, (u_m, v_m), w_1, \dots, w_{k'}$ satisfy (6), (8), (9), and we obtain $P_1[x_1], \dots, P_{m+1}[x_1]$ by (10) and (11) of Case 2 of the above nondeterministic procedure.

Summing up, every choice of $u, v, (u_1, v_1), \dots, (u_m, v_m), w_1, \dots, w_{k'}$ in our nondeterministic procedure satisfying the required conditions determines a standard sequence of k-copying one-variable tree patterns whose composition matches T, and every such sequence is determined by some choice of $u, v, (u_1, v_1), \dots, (u_m, v_m), w_1, \dots, w_{k'}$. Clearly, there are less than

$$n^2 \cdot 2^{\log_2 n} \cdot n^{k-2} = n^{k+1}$$

choices of $u, v, (u_1, v_1), \dots, (u_m, v_m), w_1, \dots, w_{k'}$. It is also clear that all these choices, as well as the corresponding standard sequences of k-copying one-variable tree patterns, can be enumerated in polynomial time. □