Distributional Learning and Context/Substructure Enumerability in Nonlinear Tree Grammars

Makoto Kanazawa[1(✉)] and Ryo Yoshinaka[2,3P]

[1] Principles of Informatics Research Division, National Institute of Informatics and SOKENDAI, Tokyo, Japan
kanazawa@nii.ac.jp
[2] Graduate School of Informatics, Kyoto University, Kyoto, Japan
[3] Graduate School of Information Sciences, Tohoku University, Sendai, Japan

Abstract. We study tree-generating almost linear second-order ACGs that admit bounded nonlinearity either on the context side or on the substructure side, and give distributional learning algorithms for them.

1 Introduction

Originally developed for efficient learning of context-free languages [3,13], the method of *distributional learning* under the paradigm of *identification in the limit from positive data and membership queries* has been successfully applied to a number of more complex grammatical formalisms that derive objects (strings, trees, λ-terms, etc.) through local sets of *derivation trees* [9,12,14]. In these formalisms, a subtree s of a complete derivation tree $t = c[s]$ contributes a certain "substructure" $S = \phi(s)$ which is contained in the whole derived object $T = \phi(t)$, and the remaining part $c[\,]$ of the derivation tree contributes a function $C = \phi(c[\,])$ that maps S to $T = C(S)$. We can think of C as a "context" that surrounds S in T. Fixing a class $\mathbb{G}$ of grammars fixes the set $\mathbb{S}$ of possible substructures and the set $\mathbb{C}$ of possible contexts that may be contributed by parts of possible derivation trees. Each language L generated by a grammar in $\mathbb{G}$ acts as an arbiter that decides which context $C \in \mathbb{C}$ should "accept" which substructure $S \in \mathbb{S}$ (i.e., whether $C(S) \in L$).

Distributional learning algorithms come in two broad varieties. In the *primal* approach, the learner first extracts all substructures and all contexts that are contained in the input data, which is a finite set of elements of the target language L_*. The learner then collects all subsets of the extracted substructures whose cardinality does not exceed a certain fixed bound m. These subsets are used as nonterminal symbols of the hypothesized grammar. Out of all possible grammar rules that can be written using these nonterminals, the learner lists those that use operations that may be involved in the generation of the objects in the input data. In the final step of the algorithm, the learner tries to validate each of these rules with the membership oracle, which answers a query "$C(S) \in L_*$?" in constant time. If a rule has a set $\mathbf{S}$ of substructures on the left-hand side and

A. Foret et al. (Eds.): FG 2015/2016, LNCS 9804, pp. 94–111, 2016.
DOI: 10.1007/978-3-662-53042-9_6

sets $\mathbf{S}_1, \dots, \mathbf{S}_r$ on the right-hand side, and the grammatical operation associated with the rule is f, then the learner determines whether the following implication holds for all contexts C extracted from the input data:

$$C(S) \in L_* \text{ for all } S \in \mathbf{S} \text{ implies} \\ C(f(S_1, \dots, S_n)) \in L_* \text{ for all } S_1 \in \mathbf{S}_1, \dots, S_n \in \mathbf{S}_n. \quad (1)$$

The grammar conjectured by the learner includes only those rules that pass this test.

The idea of the rule validation is the following: It is dictated that the elements of the nonterminal $\mathbf{S}$ together *characterize* the set of all substructures that can be derived from $\mathbf{S}$ by the hypothesized grammar in the sense that every context $C \in \mathbb{C}$ that accepts all elements of $\mathbf{S}$ must accept all substructures derived from $\mathbf{S}$. Thus, only those rules that are consistent with this requirement are allowed in the hypothesized grammar. A remarkable property of the algorithm is that it successfully learns the language of every grammar in the given class $\mathbb{G}$ that has the *m-finite kernel property* in the sense that each nonterminal is characterized by a set of substructures of cardinality up to m.

In the *dual* approach to distributional learning, the role of contexts and substructures is switched. The learner uses as nonterminals subsets of the contexts extracted from the input data with cardinality $\leq m$, and uses the extracted substructures to validate candidate rules. The algorithm learns those languages that have a grammar with the *m-finite context property* in the sense that each nonterminal is characterized by a set of contexts of cardinality $\leq m$.

Whether each of these algorithms runs in polynomial time in the size of the input data D depends on several factors that are all determined by the grammar class $\mathbb{G}$. The foremost among them is the enumeration of the two sets

$$\mathbb{S}|_D = \{\, S \in \mathbb{S} \mid C(S) \in D \text{ for some } C \in \mathbb{C} \,\}, \\ \mathbb{C}|_D = \{\, C \in \mathbb{C} \mid C(S) \in D \text{ for some } S \in \mathbb{S} \,\}.$$

There are two possible difficulties in enumerating each of these sets in polynomial time. First, the sheer number of elements of the set may be super-polynomial, in which case explicit enumeration of the set is not possible in polynomial time. Second, recognizing which substructure/context belongs to the set may be computationally costly. The second problem, even when it arises, can often be dealt with by replacing the set in question by a more easily recognizable superset without disrupting the working of the algorithm. The first problem is the more pressing one.

With all *linear* grammar formalisms to which distributional learning has been applied, neither of these two difficulties arise. When these formalisms are extended to allow nonlinearity in grammatical operations, however, the problem of super-polynomial cardinality hits hard. Thus, with *parallel multiple context-free grammars*, the nonlinear extension of *multiple context-free grammars* (successfully dealt with in [12]), the set $\mathbb{C}$ becomes a much larger set, even though $\mathbb{S}$ stays exactly the same. As a result, the cardinality of $\mathbb{C}|_D$ is no longer bounded by a polynomial. The situation with *IO context-free grammars*, the nonlinear extension of the *simple context-free tree grammars* (treated in [9]), is even worse. Both of the sets $\mathbb{S}|_D$ and $\mathbb{C}|_D$ become super-polynomial in cardinality.

When only one of the two sets $\mathbb{S}|_D$ and $\mathbb{C}|_D$ is of super-polynomial cardinality, as is the case with PMCFGs, however, there is a way out of this plight [4]. The solution is to restrict the offending set by a certain property, parametrized by a natural number, so that its cardinality will be polynomial. The parametrized restriction leads to an increasing chain of subsets inside $\mathbb{S}$ or $\mathbb{C}$. In the case of PMCFGs, we get $\mathbb{C}_1 \subset \mathbb{C}_2 \subset \mathbb{C}_3 \subset \cdots \subset \mathbb{C} = \bigcup_k \mathbb{C}_k$, where $\mathbb{C}_k$ is the set of all possible contexts that satisfy the property with respect to the parameter k. The actual property used by [4] was a measure of nonlinearity of the context ("k-copying"), but this specific choice is not crucial for the correct working of the algorithm, as long as $\mathbb{C}_k|_D$ can be enumerated in polynomial time. The learning algorithm now has two parameters, m and k: the former is a bound on the cardinality of sets of contexts the learner uses as nonterminals as before, and the latter is a restriction on the kind of context allowed in these sets. The class of languages successfully learned by the algorithm includes the languages of all grammars in the target class that have the (k, m)-*finite context-property* in the sense that each nonterminal is characterized by a subset of $\mathbb{C}_k$ of cardinality $\leq m$.

This algorithm does not learn the class of all grammars with the m-finite context property, but a proper subset of it. Nevertheless, the parametrized restriction has a certain sense of naturalness, and the resulting learnable class properly extends the corresponding linear class, so the weaker result is interesting in its own right.

In this paper, we explore the connection between distributional learning and context/substructure enumerability in the general setting of *almost linear second-order abstract categorial grammars* generating trees [5–7] ("almost linear ACGs" for short). This class of grammars properly extends IO context-free tree grammars and is equivalent in tree generating power to *tree-valued attribute grammars* [1]. In fact, the expressive power of typed lambda calculus makes it possible to faithfully encode most known tree grammars within almost linear ACGs.

Like IO context-free tree grammars and unlike PMCFGs, almost linear ACGs in general do not allow polynomial-time enumerability either on the context side or on the substructure side. Only very special grammars do, and an interesting subclass of them consists of those grammars that allow only a bounded degree of nonlinearity in the contexts (or in the substructures). It is easily decidable whether a given ACG satisfies each of these properties. We show that both of the resulting classes of grammars indeed allow a kind of efficient distributional learning similar to that for PMCFGs.

2 Typed Lambda Terms and Almost Linear ACGs

2.1 Types and Typed Lambda Terms

We assume familiarity with the notion of a *simply typed λ-term* (à la Church) over a *higher-order signature* $\Sigma = (A_\Sigma, C_\Sigma, \tau_\Sigma)$, where A_Σ is the set of *atomic* *types*, C_Σ is the set of constants, and τ_Σ is a function from C_Σ to types over A_Σ. We use standard abbreviations: $\alpha_1 \to \cdots \to \alpha_n \to p$ means $\alpha_1 \to (\cdots \to (\alpha_n \to p) \dots)$, and $\lambda x_1^{\alpha_1} \dots x_n^{\alpha_n}.MN_1 \dots N_m$ is short for $\lambda x_1^{\alpha_1}. \dots .\lambda x_n^{\alpha_n}.((\dots(MN_1)\dots)N_m)$. The *arity* of $\alpha = \alpha_1 \to \cdots \to \alpha_n \to p$ with $p \in A_\Sigma$ is $\text{arity}(\alpha) = n$. We write $\beta^n \to p$ for the type $\beta \to \cdots \to \beta \to p$ of arity n.

We take for granted such notions as β- and *η-reduction*, *β-normal form*, and *linear λ-terms*. We write $\twoheadrightarrow_\beta$ and $\twoheadrightarrow_\eta$ for the relations of β- and η-reduction between λ-terms. Every typed λ-term has a β-normal form, unique up to renaming of bound variables, which we write as $|M|_\beta$.

The set $\text{LNF}_X^\alpha(\Sigma)$ of λ-terms of type α in *η-long β-normal form* (with free variables from X) is defined inductively as follows:

– If $x^{\alpha_1 \to \cdots \to \alpha_n \to p} \in X$, $M_1 \in \text{LNF}_X^{\alpha_1}(\Sigma), \dots, M_n \in \text{LNF}_X^{\alpha_n}(\Sigma)$, and $p \in A_\Sigma$, then $x^{\alpha_1 \to \cdots \to \alpha_n \to p} M_1 \dots M_n \in \text{LNF}_X^p(\Sigma)$.
– If $c \in C_\Sigma$, $\tau_\Sigma(c) = \alpha_1 \to \cdots \to \alpha_n \to p$, $p \in A_\Sigma$, and $M_1 \in \text{LNF}_X^{\alpha_1}(\Sigma), \dots, M_n \in \text{LNF}_X^{\alpha_n}(\Sigma)$, then $cM_1 \dots M_n \in \text{LNF}_X^p(\Sigma)$.
– If $M \in \text{LNF}_{X \cup \{x^\alpha\}}^\beta(\Sigma)$, then $\lambda x^\alpha.M \in \text{LNF}_X^{\alpha \to \beta}(\Sigma)$.

We often suppress the superscript and/or subscript in $\text{LNF}_X^\alpha(\Sigma)$. Note that $\text{LNF}_\varnothing^\alpha(\Sigma)$ denotes the set of *closed* λ-terms of type α in η-long β-normal form. We note that if $M \in \text{LNF}^{\alpha \to \beta}(\Sigma)$ and $N \in \text{LNF}^\alpha(\Sigma)$, then $|MN|_\beta \in \text{LNF}^\alpha(\Sigma)$.

Henceforth, we often suppress the type superscript on variables. This is just for brevity; each variable in a typed λ-term comes with a fixed type.

We use strings over $\{0,1\}$ to refer to positions inside a λ-term or a type. We write ε for the empty string, and write $u \leq v$ to mean u is a prefix of v. When $u = u'0^i$, we refer to u' as $u0^{-i}$.

The *shape* of a type α, written $[\alpha]$, is defined by

$$[p] = \{\varepsilon\} \text{ if } p \text{ is atomic}, \qquad [\alpha \to \beta] = \{\varepsilon\} \cup \{\, 1u \mid u \in [\alpha] \,\} \cup \{\, 0u \mid u \in [\beta] \,\}.$$

The elements of $[\alpha]$ are the *positions* of α. A position u is *positive* if its parity (i.e., the number of 1s in u modulo 2) is 0, and *negative* if its parity is 1. We write $[\alpha]^+$ and $[\alpha]^-$ for the set of positive and negative positions of α, respectively. A position u of α is a *subpremise* if $u = u'1$ for some u'. Such an occurrence is a *positive* (resp. *negative*) *subpremise* if it is positive (resp. negative). We write $[\alpha]_{\text{sp}}^+$ (resp. $[\alpha]_{\text{sp}}^-$) for the set of positive (resp. negative) subpremises of $[\alpha]$.

If $u \in [\alpha]$, the *subtype* of α occurring at u, written α/u, is defined by

$$\alpha/\varepsilon = \alpha, \quad (\alpha \to \beta)/0u = \beta/u, \quad (\alpha \to \beta)/1u = \alpha/u.$$

If $\alpha/u = \beta$, we say that β *occurs* at position u in α.

Given a λ-term M, the *shape* of M, written $[M]$, is defined by

$$\begin{aligned} [M] &= \{\varepsilon\} \quad \text{if } M \text{ is a variable or a constant}, \\ [MN] &= \{\varepsilon\} \cup \{\, 0u \mid u \in [M] \,\} \cup \{\, 1u \mid u \in [N] \,\}, \\ [\lambda x.M] &= \{\varepsilon\} \cup \{\, 0u \mid u \in [M] \,\}. \end{aligned}$$

The elements of $[M]$ are the *positions* of M.

If $u \in [M]$, the *subterm* of M occurring at u, written M/u, is defined by

$$M/\varepsilon = M, \quad (MN)/0u = M/u, \quad (MN)/1u = N/u, \quad (\lambda x.M)/0u = M/u.$$

When $N = M/u$, we sometimes call u an *occurrence of* N (in M).

When $v \in [M]$ but $v0 \notin [M]$, M/v is a variable or a constant. For each $u \in [M]$, we refer to the unique occurrence of a variable or constant in $[M]$ of the form $u0^k$ as the *head* of u (in M); we also call the variable or constant occurring at the head of u the *head* of M/u.

A position $v \in [M]$ *binds* a position $u \in [M]$ if M/u is a variable x and v is the longest prefix of u such that M/v is a λ-abstract of the form $\lambda x.N$. When v binds u in M, we write $v = b_M(u)$. When every occurrence in M of a λ-abstract is the binder of some position, M is called a *λI-term*.

Let $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Sigma)$. Note that an occurrence $v \in [M]$ of a variable or a constant of type β with $\mathrm{arity}(\beta) = n$ is always accompanied by n arguments, so that $v0^{-i}$ is defined for all $i \leq n$. The set of *replaceable* occurrences [2] of bound variables in M and the negative subpremise $\mathrm{nsp}_M(u)$ of α associated with such an occurrence u, are defined as follows:[1]

(i) If $b_M(u) = 0^{j-1}$ for some $j \geq 1$ (i.e., $b_M(u)$ is the jth of the leading λs of M), then u is replaceable and $\mathrm{nsp}_M(u) = 0^{j-1}1$.

(ii) If $b_M(u) = v0^{-i}10^{j-1}$ for some replaceable v and $i, j \geq 1$ (i.e., $b_M(u)$ is the jth of the leading λs of the ith argument of v), then u is replaceable and $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)0^{i-1}10^{j-1}1$.

It is easy to see that the following conditions always hold:

– If u is a replaceable occurrence of a bound variable x^β, then $\beta = \alpha/\mathrm{nsp}_M(u)$.
– If M is a λI-term (in addition to belonging to $\mathrm{LNF}^{\alpha}_{\varnothing}(\Sigma)$), then for every $v \in [\alpha]^{-}_{\mathrm{sp}}$, there exists a $u \in [M]$ such that $\mathrm{nsp}_M(u) = v$.

Example 1. Let

$$M = \lambda y_1^o y_2^{o\to(o\to(o\to o)\to o)\to o}.y_2(fy_1a)(\lambda y_3^o y_4^{o\to o}.f(y_4(fy_3y_1))(y_4(fy_3y_1))).$$

Then $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$, where Δ contains constants f, a of type $o \to o \to o$ and o, respectively, and

$$\alpha = \overset{1}{o} \to \underbrace{(o \to \underbrace{(\overset{01011}{o} \to \underbrace{(o \to o)}_{010101} \to o)}_{0101} \to o)}_{01} \to o.$$

[1] A definition equivalent to $\mathrm{nsp}_M(u)$ for untyped λ-terms is in [2] (*access path*). The correspondence between these paths and negative subpremises for typed linear λ-terms is in [10].

- The bound variable y_1^o occurs in M at three positions, 000101, 001000111, 00100111, whose binder is ε. These positions are associated with the negative subpremise 1 in α.
- The bound variable $y_2^{o \to (o \to (o \to o) \to o) \to o}$ occurs in M at one position, 0000, whose binder is 0. This position is associated with the subpremise 01 in α.
- The bound variable y_3^o occurs in M at two positions, 0010001101 and 001001101, whose binder is 001. These positions are associated with the negative subpremise 0101.
- The bound variable $y_4^{o \to o}$ occurs in M at two positions, 00100010 and 0010010, whose binder is 0010. These positions are associated with the negative subpremise 010101.

2.2 Almost Linear Lambda Terms over a Tree Signature

Now we are going to assume that Δ is a tree signature; i.e., every constant of Δ is of type $o^r \to o$ for some $r \geq 0$, where o is the only atomic type of Δ. For a closed $M \in \mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$, every occurrence of a bound variable in M is replaceable.

A *tree* is an element of $\mathrm{LNF}^{o}_{\varnothing}(\Delta)$. A closed λ-term $M \in \mathrm{LNF}^{o^r \to o}_{\varnothing}(\Delta)$ is called a *tree context*. We say that a tree context $M = \lambda x_1 \dots x_r.N$ *matches* a tree T if there are trees $T_1, \dots, T_r$ such that $(\lambda x_1 \dots x_r.N)T_1 \dots T_r \twoheadrightarrow_\beta T$. We say that M is *contained* in T if it matches a subtree of T.

The notion of an *almost linear λ-term* was introduced by Kanazawa [5,7]. Briefly, a closed typed λ-term is almost linear if every occurrence of a λ-abstract $\lambda x^\alpha.N$ in it binds a unique occurrence of x^α, unless α is atomic, in which case it may bind more than one occurrence of x^α. Almost linear λ-terms share many of the properties of linear λ-terms; see [5–8] for details.

Almost linear λ-terms are typically not β-normal. For instance, $\lambda y^{o \to o}.(\lambda x^o.fxx)(yc)$, where f and c are constants of type $o \to o \to o$ and o, respectively, is almost linear, but its β-normal form, $\lambda y^{o \to o}.f(yc)(yc)$, is not. In this paper, we choose to deal with the η-long β-normal forms of almost linear λ-terms directly, rather than through their almost linear β-expanded forms.

We write $\mathrm{AL}^\alpha(\Delta)$ for the set of *closed* λ-terms in $\mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$ that β-expand to an almost linear λ-term. (The superscript is often omitted.) The following lemma, which we do not prove here, may be taken as the definition of $\mathrm{AL}^\alpha(\Delta)$ (see [7,8] for relevant properties of almost linear λ-terms):

Lemma 1. *Let M be a closed λI-term in $\mathrm{LNF}^{\alpha}_{\varnothing}(\Delta)$. Then $M \in \mathrm{AL}^\alpha(\Delta)$ if and only if the following conditions hold for all bound variable occurrences $u, v \in [M]$ such that $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)$, where $n = \mathrm{arity}(\alpha/\mathrm{nsp}_M(u))$:*

(i) $\{\, w \mid u0^{-n}w \in [M] \,\} = \{\, w \mid v0^{-n}w \in [M] \,\}$.
(ii) *If $M/u0^{-n}w$ is a constant, then $M/u0^{-n}w = M/v0^{-n}w$.*
(iii) *If $M/u0^{-n}w$ is a variable, then $M/v0^{-n}w$ is also a variable and $\mathrm{nsp}_M(u0^{-n}w) = \mathrm{nsp}_M(v0^{-n}w)$.*

We call $M \in \mathrm{AL}^\alpha(\Delta)$ a *canonical writing* if for all bound variable occurrences u, v of M, $\mathrm{nsp}_M(u) = \mathrm{nsp}_M(v)$ implies $M/u = M/v$ and vice versa. For example,

$\lambda y_1^{(o\to o)\to o} y_2^{(o\to o)\to o}.f(y_1(\lambda z_1^o.z_1))(y_1(\lambda z_1^o.z_1))(y_2(\lambda z_2^o.z_2))$ is a canonical writing, whereas neither $\lambda y_1^{(o\to o)\to o} y_2^{(o\to o)\to o}.f(y_1(\lambda z_1^o.z_1))(y_1(\lambda z_2^o.z_2))(y_2(\lambda z_3^o.z_3))$ nor $\lambda y_1^{(o\to o)\to o} y_2^{(o\to o)\to o}.f(y_1(\lambda z_1^o.z_1))(y_1(\lambda z_1^o.z_1))(y_2(\lambda z_1^o.z_1))$ is.

Lemma 2. *For every $M \in \mathrm{AL}^\alpha(\mathit{\Delta})$, there exists a canonical writing $M' \in \mathrm{AL}^\alpha(\mathit{\Delta})$ such that $M' \equiv_\alpha M$.*

A *pure* λ-term is a λ-term that contains no constant. We write AL^α for the subset of $\mathrm{AL}^\alpha(\mathit{\Delta})$ consisting of pure λ-terms. An important property of $\mathrm{AL}^\alpha(\mathit{\Delta})$ that we heavily rely on in what follows is that every $M \in \mathrm{AL}^\alpha(\mathit{\Delta})$ can be expressed in a unique way as an application $M^\circ M_1^\bullet \dots M_l^\bullet$ of a *pure* λ-term M° to a list of tree contexts $M_1^\bullet, \dots, M_l^\bullet$. We call the former the *container* of M and the latter its *stored tree contexts*. These λ-terms satisfy the following conditions:

1. $l \le |[\alpha]^+_{\mathrm{sp}}| + 1$,
2. $M_i^\bullet \in \mathrm{AL}^{o^{r_i}\to o}(\mathit{\Delta})$ for some $r_i \le |[\alpha]^-_{\mathrm{sp}}|$ for each $i = 1, \dots, l$,
3. $M^\circ \in \mathrm{AL}^{(o^{r_1}\to o)\to\dots\to(o^{r_l}\to o)\to\alpha}$,
4. $M^\circ M_1^\bullet \dots M_l^\bullet \twoheadrightarrow_\beta M$.

The formal definition of this separation of $M \in \mathrm{AL}^\alpha(\mathit{\Delta})$ into its container and stored tree contexts is rather complex, but the intuitive idea is quite simple. The stored tree contexts of M are the maximal tree contexts that can be discerned in the input λ-term.

Example 2. Consider the λ-term M of type $\alpha = o\to(o\to(o\to(o\to o)\to o)\to o)\to o$ in Example 1. This λ-term belongs to $\mathrm{AL}^\alpha(\mathit{\Delta})$. Its container and stored tree contexts are:

$$M^\circ = \lambda z_1^{o\to o} z_2^{o\to o} z_3^{o\to o\to o} y_1^o y_2^{o\to(o\to(o\to o)\to o)\to o}.y_2(z_1 y_1)(\lambda y_3^o y_4^{o\to o}.z_2(y_4(z_3 y_3 y_1))),$$
$$M_1^\bullet = \lambda x_1.f x_1 a, \quad M_2^\bullet = \lambda x_1.f x_1 x_1, \quad M_3^\bullet = \lambda x_1 x_2.f x_1 x_2.$$

Here is the formal definition. Let $M \in \mathrm{AL}^\alpha(\mathit{\Delta})$. We assume that M is canonical. Then $|[\alpha]^-_{\mathrm{sp}}|$ is exactly the number of distinct bound variables in M. Let $s_1, \dots, s_k$ list the elements of $[\alpha]^-_{\mathrm{sp}}$ in lexicographic order. Let $y_1, \dots, y_k$ be the corresponding list of bound variables in M, and let $n_i = \mathrm{arity}(\alpha/s_i)$ for each $i = 1, \dots, k$. Note that

$$\sum_{i=1}^k n_i \le |[\alpha]^+_{\mathrm{sp}}|.$$

The canonicity of M implies that every occurrence of y_i in M is accompanied by the exact same list of arguments $N_{i,1}, \dots, N_{i,n_i}$. The type of $N_{i,j}$ is $\alpha/s_i 0^{j-1} 1$.

Let $x_1, \dots, x_k$ be fresh variables of type o. For each subterm N of M of type o, define $N^\blacktriangle$ by

$$(cT_1 \dots T_n)^\blacktriangle = cT_1^\blacktriangle \dots T_n^\blacktriangle, \quad (y_i N_{i,1} \dots N_{i,n_i})^\blacktriangle = x_i.$$

Let M' be the maximal subterm of M of atomic type; in other words, M' is the result of stripping M of its leading λs. Likewise, let $N'_{i,j}$ be the maximal subterm of $N_{i,j}$ of atomic type. Let $(M_1, \dots, M_l)$ be the sublist of

$$(M', N'_{1,1}, \dots, N'_{1,n_1}, \dots, N'_{k,1}, \dots, N'_{k,n_k})$$

consisting of the λ-terms whose head is a constant. (This list will contain duplicates if there exist i_1, j_1, i_2, j_2 such that $(i_1, j_1) \neq (i_2, j_2)$, $N'_{i_1,j_1} = N'_{i_2,j_2}$, and the head of this λ-term is a constant.) For each $i = 1, \ldots, l$, let $x_{m_{i,1}}, \ldots, x_{m_{i,r_i}}$ list the variables in $M_i^{\blacktriangle}$, in the order of their first appearances in $M_i^{\blacktriangle}$. Define

$$M_i^{\bullet} = \lambda x_{m_{i,1}} \ldots x_{m_{i,r_i}}.M_i^{\blacktriangle}, \quad \overrightarrow{M^{\bullet}} = (M_1^{\bullet}, \ldots, M_l^{\bullet}).$$

These are the stored tree contexts of M.

In order to define the container M°, we first define $N^{\triangle}$ by induction for each subterm N of M that is either (i) some M_i, (ii) a λ-term of atomic type whose head is a variable, or (iii) a λ-abstract. Let $z_1, \ldots, z_l$ be fresh variables of type $o^{r_1} \to o, \ldots, o^{r_l} \to o$, respectively[2].

$$\begin{aligned} M_i^{\triangle} &= z_i(y_{m_{i,1}} N_{m_{i,1},1} \ldots N_{m_{i,1},n_{m_{i,1}}})^{\triangle} \ldots (y_{m_{i,r_i}} N_{m_{i,r_i},1} \ldots N_{m_{i,r_i},n_{m_{i,r_i}}})^{\triangle}, \\ (y_i N_{i,1} \ldots N_{i,n_i})^{\triangle} &= y_i N_{i,1}^{\triangle} \ldots N_{i,n_i}^{\triangle}, \\ (\lambda y_i.N)^{\triangle} &= \lambda y_i.N^{\triangle}. \end{aligned}$$

Finally, define

$$M^{\circ} = \lambda z_1 \ldots z_l.M^{\triangle}.$$

Lemma 3. *$M^{\circ}, \overrightarrow{M^{\bullet}}$ satisfy the required conditions.*

Lemma 4. *Let $N \in \mathrm{AL}^{\alpha_1 \to \cdots \to \alpha_n \to \beta}(\varDelta), M_i \in \mathrm{AL}^{\alpha_i}(\varDelta)$ $(i = 1, \ldots, n)$, and $P = |NM_1 \ldots M_n|_{\beta} \in \mathrm{AL}^{\beta}(\varDelta)$. Suppose*

$$\overrightarrow{M_i^{\bullet}} = ((M_i)_1^{\bullet}, \ldots, (M_i)_{l_i}^{\bullet}), \quad (M_i)_j^{\bullet} \in \mathrm{AL}^{o^{r_{i,j}} \to o}(\varDelta), \qquad \overrightarrow{P^{\bullet}} = (P_1^{\bullet}, \ldots, P_m^{\bullet}).$$

For $i = 1, \ldots, n$ and $j = 1, \ldots, l_i$, let $c_{i,j}$ be a fresh constant of type $o^{r_{i,j}} \to o$. Let $\varDelta'$ be the tree signature that extends $\varDelta$ with the $c_{i,j}$, and let

$$Q = |N((M_1)^{\circ} c_{1,1} \ldots c_{1,l_1}) \ldots ((M_n)^{\circ} c_{n,1} \ldots c_{n,l_n})|_{\beta}.$$

We can compute the container and stored tree contexts of $Q \in \mathrm{AL}^{\beta}(\varDelta')$ with respect to $\varDelta'$. Then we have

$$P^{\circ} = Q^{\circ}, \quad P_i^{\bullet} = |(Q_i^{\bullet})[c_{i,j} := (M_i)_j^{\bullet}]|_{\beta},$$

where $[c_{i,j} := (M_i)_j^{\bullet}]$ denotes the substitution of $(M_i)_j^{\bullet}$ for each $c_{i,j}$.

Definition 1. Let $M \in \mathrm{AL}^{\alpha}(\varDelta)$.

(i) The *unlimited profile* of M is $\mathrm{prof}_{\infty}(M) = (M^{\circ}, w_1, \ldots, w_l)$, where l is the length of $\overrightarrow{M^{\bullet}} = (M_1^{\bullet}, \ldots, M_l^{\bullet})$ and for each i, w_i is the r_i-tuple of positive integers whose jth component is the number of occurrences of the jth bound variable in $M_i^{\bullet}$.

[2] When $M_i = M_j$ for some distinct i, j, the definition of $M_i^{\triangle}$ in fact depends on the subscript i.

(ii) For $k \geq 1$, the *k-threshold profile* of M, written $\mathrm{prof}_k(M)$, is just like its unlimited profile except that any number greater than k is replaced by ∞.

The *type* of the (unlimited or k-threshold) profile of M is α.

Example 3. The unlimited profile of the λ-term M from Example 1 is $\mathrm{prof}(M) = (M^\circ, (1), (2), (1, 1))$. Its 1-threshold profile is $\mathrm{prof}_1(M) = (M^\circ, (1), (\infty), (1, 1))$, and its k-threshold profile for $k \geq 2$ is the same as its unlimited profile.

Lemma 5. *For each $k \geq 1$ and type α, there are only finitely many k-threshold profiles of type α.*

We say that a k-threshold profile $(M^\circ, w_1, \dots, w_l)$ is *k-bounded* if $w_i \in \{1, \dots, k\}^{r_i}$ for $i = 1, \dots, l$. A λ-term $M \in \mathrm{AL}(\Delta)$ that has a k-bounded profile is called *k-bounded.* We write $\mathrm{AL}_k^\alpha(\Delta)$ for the set of all k-bounded λ-terms in $\mathrm{AL}^\alpha(\Delta)$.

Note that $M \in \mathrm{AL}(\Delta)$ is linear if and only if it is 1-bounded and has a linear container.

Lemma 6. *Let $N \in \mathrm{AL}^{\alpha_1 \to \dots \to \alpha_n \to \beta}(\Delta)$, and $M_i, M_i' \in \mathrm{AL}^{\alpha_i}(\Delta)$ for each $i = 1, \dots, n$. Suppose that for each $i = 1, \dots, n$, $\mathrm{prof}_k(M_i) = \mathrm{prof}_k(M_i')$. Then $\mathrm{prof}_k(|N M_1 \dots M_n|_\beta) = \mathrm{prof}_k(|N M_1' \dots M_n'|_\beta)$.*

The above lemma justifies the notation $N\pi_1 \dots \pi_n$ for $\mathrm{prof}_k(|N M_1 \dots M_n|_\beta)$ with $\mathrm{prof}_k(M_i) = \pi_i$, when k is understood from context. When $N = \lambda x_1 \dots x_n.Q$, we may also write $Q[x_1 := \pi_1, \dots, x_n := \pi_n]$ for $N\pi_1 \dots \pi_n$. In this way, we can freely write profiles in expressions that look like λ-terms, like $\lambda x.\pi_1(Mx\pi_2)$.

Lemma 7. *Given a λ-term $N \in \mathrm{AL}^{\alpha_1 \to \dots \to \alpha_n \to \beta}(\Delta)$ and k-threshold profiles $\pi_1, \dots, \pi_n$ of type $\alpha_1, \dots, \alpha_n$, respectively, the k-threshold profile $N\pi_1 \dots \pi_n$ can be computed in polynomial time.*

In what follows, we often speak of "profiles" to mean k-threshold profiles, letting the context determine the value of k.

2.3 Almost Linear Second-Order ACGs on Trees

A (tree-generating) *almost linear second-order ACG* $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ consists of a second-order signature Σ (*abstract vocabulary*), a tree signature Δ (*object vocabulary*), a set $\mathscr{I} \subseteq A_\Sigma$ of *distinguished types*, and a *higher-order homomorphism* $\mathcal{H}$ that maps each atomic type $p \in A_\Sigma$ to a type $\mathcal{H}(p)$ over A_Δ and each constant $c \in C_\Sigma$ to its *object realization* $\mathcal{H}(c) \in \mathrm{AL}^{\mathcal{H}(\tau_\Delta(c))}(\Delta)$. It is required that the image of $\mathscr{I}$ under $\mathcal{H}$ is $\{o\}$. That Σ is second-order means that for every $c \in C_\Sigma$, its type $\tau_\Sigma(c)$ is of the form $p_1 \to \dots \to p_n \to q$; thus, any λ-term in $\mathrm{LNF}^p_\varnothing(\Sigma)$ for $p \in A_\Sigma$ has the form of a tree. A closed *abstract term* $P \in \mathrm{LNF}^\alpha_\varnothing(\Sigma)$ is homomorphically mapped by $\mathcal{H}$ to its object realization $|\mathcal{H}(P)|_\beta \in \mathrm{AL}^{\mathcal{H}(\alpha)}(\Delta)$. For $p \in A_\Sigma$, we write $\mathcal{S}(\mathscr{G}, p)$ for $\{\, |\mathcal{H}(P)|_\beta \mid P \in \mathrm{LNF}^p_\varnothing(\Sigma) \,\}$ and

$\mathcal{C}(\mathscr{G},p)$ for $\{\, |\mathcal{H}(Q)|_\beta \mid Q$ is a closed linear λ-term in $\mathrm{LNF}^{p\to s}_{\varnothing}(\Sigma)$ for some $s \in \mathscr{I}\,\}$. The elements of these sets are *substructures* and *contexts* of $\mathscr{G}$, respectively. The tree language generated by $\mathscr{G}$ is $\mathcal{O}(\mathscr{G}) = \bigcup_{s\in\mathscr{I}} \mathcal{S}(\mathscr{G},s)$.

An abstract constant $c \in C_\Sigma$ together with its type $\tau(c)$ and its object realization $\mathcal{H}(c)$ corresponds to a rule in more traditional grammar formalisms. An abstract atomic type $p \in A_\Sigma$ corresponds to a nonterminal. We say that $\mathscr{G}$ is *rule-k-bounded* if $\mathcal{H}(c)$ is k-bounded for every abstract constant $c \in C_\Sigma$.

Definition 2. Let $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ be a tree-generating almost linear second-order ACG.

(i) We say that $\mathscr{G}$ is *substructure-k-bounded* if $\mathcal{S}(\mathscr{G},p) \subseteq \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ for all atomic types $p \in A_\Sigma$.
(ii) We say that $\mathscr{G}$ is *context-k-bounded* if $\mathcal{C}(\mathscr{G},p) \subseteq \mathrm{AL}_k^{\mathcal{H}(p)\to o}(\Delta)$ for all atomic types $p \in A_\Sigma$.

The set of possible k-threshold profiles of elements of $\mathcal{S}(\mathscr{G},p)$ or $\mathcal{C}(\mathscr{G},p)$ can easily be computed thanks to Lemmas 5 and 6, so substructure-k-boundedness and context-k-boundedness are both decidable properties of almost linear second-order ACGs. Conversely, one can design a substructure-k-bounded almost linear ACG by first assigning to each $p \in A_\Sigma$ a possible profile set Π_p consisting of profiles of type $\mathcal{H}(p)$; then, as the realization $\mathcal{H}(c)$ of a constant c of type $p_1 \to \cdots \to p_n \to q$, we admit only λ-terms in $\mathrm{AL}_k^{\mathcal{H}(p_1\to\cdots\to p_n\to q)}(\Delta)$ that satisfy

$$\mathcal{H}(c)\Pi_{p_1}\dots\Pi_{p_n} \subseteq \Pi_q, \tag{2}$$

where $M\Pi_1\dots\Pi_n = \{\, M\pi_1\dots\pi_n \mid \pi_i \in \Pi_i \ (i = 1,\dots,n)\,\}$. To construct a context-k-bounded almost linear ACG, we need to assign a possible context profile set Ξ_p in addition to Π_p to each $p \in A_\Sigma$. The realization $\mathcal{H}(c)$ must satisfy

$$\lambda x.\Xi_q(\mathcal{H}(c)\Pi_{p_1}\dots\Pi_{p_{i-1}}x\Pi_{p_{i+1}}\dots\Pi_{p_n}) \subseteq \Xi_{p_i} \tag{3}$$

for all $i = 1,\dots,n$, in addition to (2). Note that (2) and (3) are "local" properties of rules of ACGs. Instead of Definition 2, one may take this local constraint as a definition of substrucure/context-k-bounded almost linear ACGs.

Example 4. Let $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$, where $A_\Sigma = \{p_1, p_2, s\}$, $C_\Sigma = \{a, b, c_1, c_2, d_1, d_2\}$, $\tau_\Sigma(a) = p_1 \to s$, $\tau_\Sigma(b) = p_2 \to p_1$, $\tau_\Sigma(c_i) = p_i \to p_i$, $\tau_\Sigma(d_i) = p_i$, $A_\Delta = \{o\}$, $C_\Delta = \{e, f\}$, $\tau_\Delta(f) = o \to o \to o$, $\tau_\Delta(e) = o$, $\mathscr{I} = \{s\}$, $\mathcal{H}(p_i) = (o \to o) \to o \to o$, $\mathcal{H}(s) = o$ and

$$\begin{aligned}
\mathcal{H}(a) &= \lambda x^{(o\to o)\to o\to o}.x(\lambda z^o.z)e,\\
\mathcal{H}(b) &= \lambda x^{(o\to o)\to o\to o}y^{o\to o}z^o.x(\lambda w^o.y(fww))z,\\
\mathcal{H}(c_i) &= \lambda x^{(o\to o)\to o\to o}y^{o\to o}z^o.x(\lambda w^o.yw)(fzz),\\
\mathcal{H}(d_i) &= \lambda y^{o\to o}z^o.y(fzz)\,.
\end{aligned}$$

This grammar is rule-2-bounded and generates the set of perfect binary trees of height $\geq$ 1. We have, for example, $\mathcal{H}(b(c_2 d_2)) \in \mathcal{S}(\mathscr{G}, p_1)$ and $\mathcal{H}(\lambda x^{p_2}.a(c_1(b(c_2 x)))) \in \mathcal{C}(\mathscr{G}, p_2)$, and

$$\begin{aligned} |\mathcal{H}(b(c_2 d_2))|_\beta &= \lambda y^{o \to o} z^o . y(f(f(fzz)(fzz))(f(fzz)(fzz))), \\ |\mathcal{H}(\lambda x^{p_2}.a(c_1(b(c_2 x))))|_\beta &= \lambda x^{(o \to o) \to o \to o}.x(\lambda z.fzz)(f(fee)(fee)). \end{aligned}$$

One can see

$$\mathrm{prof}_\infty(\mathcal{S}(\mathscr{G}, p_1)) = \mathrm{prof}_\infty(\mathcal{S}(\mathscr{G}, p_2)) = \{ (\lambda {z_1}^{o \to o} y^{o \to o} w^o . y(z_1 w), (2^n)) \mid n \geq 1 \},$$

and

$$\begin{aligned} \mathrm{prof}_\infty(\mathcal{C}(\mathscr{G}, p_1)) &= \{ (\lambda {z_1}^o x^{(o \to o) \to o \to o}.x(\lambda w^o.w) z_1, ()) \}, \\ \mathrm{prof}_\infty(\mathcal{C}(\mathscr{G}, p_2)) &= \{ (\lambda {z_1}^o x^{(o \to o) \to o \to o}.x(\lambda w^o.w) z_1, ()), \\ &\qquad (\lambda {z_1}^{o \to o} {z_2}^o x^{(o \to o) \to o \to o}.x(\lambda w^o.z_1 w) z_2, (2), ()) \}. \end{aligned}$$

The grammar is context-2-bounded, but not substructure-k-bounded for any k. If a new constant a' of type $p_1 \to s$ with $\mathcal{H}(a') = \lambda x^{(o \to o) \to o \to o}.x(\lambda z^o.fzz)e$ is added to $\mathscr{G}$, the grammar is not context-2-bounded any more, since $|\mathcal{H}(\lambda x^{p_2}.a'(bx))|_\beta = \lambda x^{(o \to o) \to o \to o}.x(\lambda z^o.f(fzz)(fzz))e \in \mathcal{C}(\mathscr{G}, p_2)$.

3 Extraction of Tree Contexts from Trees

We say that $M \in \mathrm{AL}^\alpha(\Delta)$ is *contained* in a tree T if there is an $N \in \mathrm{AL}^{\alpha \to o}(\Delta)$ such that $NM \twoheadrightarrow_\beta T$. The problem of extracting λ-terms in $\mathrm{AL}^\alpha(\Delta)$ contained in a given tree reduces to the problem of extracting tree contexts from trees.

Explicitly enumerating all tree contexts of type $o^r \to o$ is clearly intractable. A perfect binary tree with n leaves (labeled by the same constant) contains more than 2^n tree contexts of type $o \to o$.

It is easy to explicitly enumerate all tree contexts of type $o^r \to o$ that are *k-copying* in the sense that each bound variable occurs at most k times. (Just pick at most $rk + 1$ nodes to determine such a tree context.) Hence it is easy to explicitly enumerate all $M \in \mathrm{AL}_k^\alpha(\Delta)$ whose stored tree contexts (which are all k-copying) are contained in a given tree. (Recall that there is a fixed finite set of candidate containers for each α.) Not all these λ-terms are themselves contained in T, but it is harmless and simpler to list them all than to enumerate exactly those λ-terms $M \in \mathrm{AL}_k^\alpha(\Delta)$ for which there is an $N \in \mathrm{AL}^{\alpha \to o}(\Delta)$ (which may not be k-bounded) such that $MN \twoheadrightarrow_\beta T$.

We consider distributional learners for tree-generating almost linear second-order ACGs who are capable of extracting k-copying tree contexts from trees. Such a learner conjectures rule-k-bounded almost linear ACGs, and use only k-bounded substructures and k-bounded contexts in order to form hypotheses.

4 Distributional Learning of One-Side k-bounded ACGs

We present two distributional learning algorithms, a primal one for the context-k-bounded almost linear ACGs, and a dual one for the substructure-k-bounded almost linear ACGs.

In distributional learning, we often have to fix certain parameters that restrict the class $\mathbb{G}$ of grammars available to the learner as possible hypotheses, in order to make the universal membership problem solvable in polynomial time. This is necessary since the learner needs to check whether the previous conjecture generates all the positive examples received so far, including the current one. In the case of almost linear ACGs, the parameters are the maximal arity n of the type of abstract constants and the finite set Ω of the possible object images of abstract atomic types. When these parameters are fixed, the universal membership problem "$T \in \mathcal{O}(\mathscr{G})$?" is in P [7].

In addition to these two parameters, we also fix a positive integer k so that any hypothesized grammar is rule-k-bounded, for the reason explained in the previous section. The hypothesis space for our learners is thus determined by three parameters, Ω, n, k. We write $\mathbb{G}(\Omega, n, k)$ for the class of grammars determined by these parameters.

In what follows, we often use sets of profiles or λ-terms inside expressions that look like λ-terms, as we did in (2) and (3) in Sect. 2.3.

4.1 Learning Context-k-bounded ACGs with the Finite Kernel Property

For $\mathbf{T} \subseteq \mathrm{LNF}^{o}_{\varnothing}(\Delta)$ and $\mathbf{R} \subseteq \mathrm{AL}^{\alpha}(\Delta)$, we define the *$k$-bounded context set of* $\mathbf{R}$ *with respect to* $\mathbf{T}$ by

$$\mathrm{Con}_k(\mathbf{T}|\mathbf{R}) = \{\, Q \in \mathrm{AL}_k^{\alpha \to o}(\Delta) \mid |QR|_\beta \in \mathbf{T} \text{ for all } R \in \mathbf{R} \,\}\,.$$

Definition 3. A context-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have the *profile-insensitive* (k, m)*-finite kernel property* if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{S}_p \subseteq \mathcal{S}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ such that $|\mathbf{S}_p| \leq m$ and

$$\mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) = \mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)).$$

This may be thought of as a primal analogue of the notion of (k, m)-FCP in [4] for the present case. It turns out, however, designing a distributional learning algorithm targeting grammars satisfying this definition is neither elegant nor quite as straightforward as existing distributional algorithms. One reason is that simply validating hypothesized rules against k-bounded contexts (see (1) in Sect. 1) does not produce a context-k-bounded grammar. Recall that to construct a context-k-bounded grammar, we must fix an assignment of an admissible substructure profile set Π_p and an admissible context profile set Ξ_p to each atomic type p which restricts the object realizations of abstract constants of each type.

We let our learning algorithm use such an assignment together with finite sets of k-bounded substructures in constructing grammar rules, and make the validation of rules sensitive to the context profile set assigned to the "left-hand side" nonterminal. This naturally leads to the following definition:

Definition 4. A context-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have the *profile-sensitive* (k,m)*-finite kernel property* ((k,m)-FKP$_{\text{prof}}$) if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{S}_p \subseteq \mathcal{S}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p)}(\Delta)$ such that $|\mathbf{S}_p| \leq m$ and

$$\mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) \cap \mathrm{prof}_k^{-1}(\Xi) = \mathrm{Con}_k(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)) \cap \mathrm{prof}_k^{-1}(\Xi)\,, \qquad (4)$$

where $\Xi = \mathrm{prof}_k(\mathcal{C}(\mathscr{G}, p))$. Such a set $\mathbf{S}_p$ is called a *characterizing substructure set of* p.

Clearly, if a context-k-bounded grammar satisfies Definition 3, then it satisfies the (k,m)-FKP$_{\text{prof}}$, so the class of grammars with (k,m)-FKP$_{\text{prof}}$ is broader than the class given by Definition 3. The notion of (k,m)-FKP$_{\text{prof}}$ is also monotone in k in the sense that (4) implies

$$\mathrm{Con}_{k+1}(\mathcal{O}(\mathscr{G})|\mathbf{S}_p) \cap \mathrm{prof}_{k+1}^{-1}(\Xi') = \mathrm{Con}_{k+1}(\mathcal{O}(\mathscr{G})|\mathcal{S}(\mathscr{G}, p)) \cap \mathrm{prof}_{k+1}^{-1}(\Xi')\,,$$

where $\Xi' = \mathrm{prof}_{k+1}(\mathcal{C}(\mathscr{G}, p)) = \mathrm{prof}_k(\mathcal{C}(\mathscr{G}, p))$, as long as $\mathscr{G}$ is context-k-bounded. This means that as we increase the parameter k, the class of grammars satisfying (k,m)-FKP$_{\text{prof}}$ monotonically increases. This is another advantage of Definition 4 over Definition 3.

The polynomial enumerability of the k-bounded λ-terms makes an efficient primal distributional learner possible for the class of context-k-bounded grammars in $\mathbb{G}(\Omega, n, k)$ with the (k,m)-FKP$_{\text{prof}}$.

Algorithm. Hereafter we fix a learning target $\mathbf{T}_* \subseteq \mathrm{LNF}_\varnothing^o(\Delta)$ which is generated by $\mathscr{G}_* = (\Sigma, \Delta, \mathcal{H}, \mathscr{I}) \in \mathbb{G}(\Omega, n, k)$ with the (k,m)-FKP$_{\text{prof}}$. We write $\mathbf{S}^{[\Xi]} = \mathrm{Con}_k(\mathbf{T}_*|\mathbf{S}) \cap \mathrm{prof}_k^{-1}(\Xi)$ for a k-bounded profile set Ξ.

For a tree $T \in \mathrm{LNF}_\varnothing^o(\Delta)$, let $\mathrm{Ext}_k^\alpha(T) = \{\, M \in \mathrm{AL}_k^\alpha(\Delta) \mid \overrightarrow{M^\bullet}$ are contained in $T\,\}$. Define

$$\begin{aligned}
\mathrm{Sub}_k^\Omega(\mathbf{D}) &= \bigcup \{\, \mathrm{Ext}_k^\alpha(T) \mid T \in \mathbf{D},\ \alpha \in \Omega\,\},\\
\mathrm{Glue}_k^{\Omega,n}(\mathbf{D}) &= \bigcup \{\, \mathrm{Ext}_k^{\alpha_1 \to \cdots \to \alpha_j \to \alpha_0}(T) \mid T \in \mathbf{D},\ \alpha_i \in \Omega \text{ for } i = 1, \dots, j\\
&\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{and } j \leq n\,\},\\
\mathrm{Con}_k^\Omega(\mathbf{D}) &= \bigcup \{\, \mathrm{Ext}_k^{\alpha \to o}(T) \mid T \in \mathbf{D},\ \alpha \in \Omega\,\}.
\end{aligned}$$

It is easy to see that $\mathcal{H}(c) \in \mathrm{Glue}_k^{\Omega,n}(\mathbf{T}_*)$ for all $c \in C_\Sigma$.

Our learner (Algorithm 1) constructs a context-k-bounded ACG $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F}) = (\Gamma, \Delta, \mathcal{J}, \mathscr{J})$ from three sets $\mathbf{K} \subseteq \mathrm{Sub}_k^\Omega(\mathbf{D})$, $\mathbf{B} \subseteq \mathrm{Glue}_k^{\Omega,n}(\mathbf{D})$ and $\mathbf{F} \subseteq \mathrm{Con}_k^\Omega(\mathbf{D})$, where $\mathbf{D}$ is a finite set of positive examples given to the

Algorithm 1. Learning ACGs in $\mathbb{G}(\Omega, n, k)$ with the (k, m)-FKP$_{\text{prof}}$.

```
Data: A positive presentation $T_1, T_2, \dots$ of $\mathbf{T}_*$; membership oracle on $\mathbf{T}_*$;
Result: A sequence of ACGs $\mathscr{G}_1, \mathscr{G}_2, \dots$;
let $\mathbf{D} := \mathbf{K} := \mathbf{B} := \mathbf{F} := \varnothing$; $\hat{\mathscr{G}} := \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$;
for $i = 1, 2, \dots$ do
  let $\mathbf{D} := \mathbf{D} \cup \{T_i\}$; $\mathbf{F} := \mathrm{Con}_k^{\Omega}(\mathbf{D})$;
  if $\mathbf{D} \not\subseteq \mathcal{O}(\hat{\mathscr{G}})$ then
    let $\mathbf{B} := \mathrm{Glue}_k^{\Omega,n}(\mathbf{D})$;
    let $\mathbf{K} := \mathrm{Sub}_k^{\Omega}(\mathbf{D})$;
  end if
  output $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ as $\mathscr{G}_i$;
end for
```

learner. As with previous primal learning algorithms, whenever we get a positive example that is not generated by our current conjecture, we expand $\mathbf{K}$ and $\mathbf{B}$, while in order to suppress incorrect rules, we keep expanding $\mathbf{F}$.

Each abstract atomic type of our grammar is a triple of a subset of $\mathbf{K}$, a k-threshold profile set, and a k-bounded profile set:

$$\begin{aligned} A_\Gamma = \{\, [\![\mathbf{S}, \Pi, \Xi]\!] \mid \mathbf{S} \subseteq \mathbf{K} \cap \mathrm{prof}_k^{-1}(\Pi) \text{ with } 1 \le |\mathbf{S}| \le m, \text{ where for some } \alpha \in \Omega,\\ \Pi \text{ is a set of } k\text{-threshold profiles of type } \alpha \text{ and}\\ \Xi \text{ is a set of } k\text{-bounded profiles of type } \alpha \to o \,\}\,. \end{aligned}$$

We have $|A_\Gamma| \le 2^{2\ell} |\mathbf{K}|^m$, where ℓ is the total number of profiles of relevant types, which is a constant.

The set of distinguished types is defined as

$$\mathscr{J} = \{\, [\![\, \mathbf{S},\ \{(\lambda z^o.z)\},\ \{(\lambda y^o.y)\} \,]\!] \in A_\Gamma \mid \mathbf{S} \subseteq \mathbf{T}_* \,\}\,,$$

which is determined by membership queries. Define $\mathcal{J}([\![\mathbf{S}, \Pi, \Xi]\!])$ to be the type of the profiles in Π.

We have an abstract constant $d \in C_\Gamma$ such that

$$\begin{aligned} &\tau_\Gamma(d) = [\![\mathbf{S}_1, \Pi_1, \Xi_1]\!] \to \cdots \to [\![\mathbf{S}_j, \Pi_j, \Xi_j]\!] \to [\![\mathbf{S}_0, \Pi_0, \Xi_0]\!] \text{ with } j \le n\,,\\ &\mathcal{J}(d) = R \in \mathbf{B}\,, \end{aligned}$$

if

- $R\Pi_1 \dots \Pi_j \subseteq \Pi_0$,
- $\lambda x.\Xi_0(R\Pi_1 \dots \Pi_{i-1} x \Pi_{i+1} \dots \Pi_j) \subseteq \Xi_i$ for $i = 1, \dots, j$,
- $|Q(RS_1 \dots S_j)|_\beta \in \mathbf{T}_*$ for all $Q \in \mathbf{S}_0^{[\Xi_0]} \cap \mathbf{F}$ and $S_i \in \mathbf{S}_i$ for $i = 1, \dots, j$.

The last condition is checked with the aid of the membership oracle.

Lemma 8. *We have* $\mathrm{prof}_k(N) \in \Pi$ *for all* $N \in \mathcal{S}(\mathscr{G}, [\![\mathbf{S}, \Pi, \Xi]\!])$*, and* $\mathrm{prof}_k(M) \in \Xi$ *for all* $M \in \mathcal{C}(\mathscr{G}, [\![\mathbf{S}, \Pi, \Xi]\!])$*. The grammar* $\mathcal{G}(\mathbf{K}, \mathbf{B}, \mathbf{F})$ *is context-k-bounded.*

Lemma 9.
If $\mathbf{K} \subseteq \mathbf{K}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K}',\mathbf{B},\mathbf{F}))$.
If $\mathbf{B} \subseteq \mathbf{B}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B}',\mathbf{F}))$.
If $\mathbf{F} \subseteq \mathbf{F}'$*, then* $\mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})) \supseteq \mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F}'))$.

Lemma 10. *Let* $\mathbf{S}_p$ *be a characterizing set of each atomic type* $p \in A_\Sigma$ *of the target grammar* $\mathscr{G}_*$*. Then* $\mathbf{S}_p \subseteq \mathrm{Sub}_k^{\Omega}(\mathbf{T}_*)$*. Moreover, if* $\mathbf{S}_p \subseteq \mathbf{K}$ *for all* $p \in A_\Sigma$ *and* $\mathcal{H}(c) \in \mathbf{B}$ *for all* $c \in C_\Sigma$*, then* $\mathbf{T}_* \subseteq \mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F}))$ *for any* $\mathbf{F}$.

We say that an abstract constant d of type $[\![\mathbf{S}_1, \Pi_1, \Xi_1]\!] \to \cdots \to [\![\mathbf{S}_j, \Pi_j, \Xi_j]\!] \to [\![\mathbf{S}_0, \Pi_0, \Xi_0]\!]$ is *invalid* if $|Q(\mathcal{J}(c)S_1 \dots S_j)|_\beta \notin \mathbf{T}_*$ for some $Q \in \mathbf{S}_0^{[\Xi_0]}$ and $S_i \in \mathbf{S}_i$.

Lemma 11. *For every* $\mathbf{K}$ *and* $\mathbf{B}$*, there is a finite set* $\mathbf{F} \subseteq \mathrm{Con}_k^{\Omega}(\mathbf{T}_*)$ *of cardinality* $|\mathbf{B}||A_\Gamma|^{n+1}$ *such that* $\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})$ *has no invalid constant.*

Lemma 12. *If* $\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})$ *has no invalid constant, then* $\mathcal{O}(\mathcal{G}(\mathbf{K},\mathbf{B},\mathbf{F})) \subseteq \mathbf{T}_*$.

Theorem 1. *Algorithm 1 successfully learns all grammars in* $\mathbb{G}(\Omega, n, k)$ *with the* (k,m)-$\mathrm{FKP}_{\mathrm{prof}}$.

We remark on the efficiency of our algorithm. It is easy to see that the description sizes of $\mathbf{K}$ and $\mathbf{B}$ are polynomially bounded by that of $\mathbf{D}$, and so is that of Γ. We need at most a polynomial number of membership queries to construct a grammar. Thus Algorithm 1 updates its conjecture in polynomial time in $\|\mathbf{D}\|$. Moreover, we do not need too much data. To make $\mathbf{K}$ and $\mathbf{B}$ satisfy the condition of Lemma 10, $m|A_\Sigma|+|C_\Sigma|$ examples are enough. To remove invalid constants, polynomially many contexts are enough by Lemma 11.

4.2 Learning Substructure-k-bounded ACGs with the Finite Context Property

For sets $\mathbf{T} \subseteq \mathrm{LNF}_\varnothing^o(\Delta)$ and $\mathbf{Q} \subseteq \mathrm{AL}_k^{\alpha \to o}(\Delta)$, we define the *k-bounded substructure set of* $\mathbf{Q}$ *with respect to* $\mathbf{T}$ by

$$\mathrm{Sub}_k(\mathbf{T}|\mathbf{Q}) = \{\, R \in \mathrm{AL}_k^{\alpha}(\Delta) \mid |QR|_\beta \in \mathbf{T} \text{ for all } Q \in \mathbf{Q} \,\}.$$

Again, we target grammars that satisfy a property sensitive to profile sets assigned to nonterminals:

Definition 5. A substructure-k-bounded ACG $\mathscr{G} = (\Sigma, \Delta, \mathcal{H}, \mathscr{I})$ is said to have *the profile-sensitive* (k,m)*-finite context property* ((k,m)-$\mathrm{FCP}_{\mathrm{prof}}$) if for every abstract atomic type $p \in A_\Sigma$, there is a nonempty set $\mathbf{Q}_p \subseteq \mathcal{C}(\mathscr{G}, p) \cap \mathrm{AL}_k^{\mathcal{H}(p) \to o}(\Delta)$ of k-bounded λ-terms such that $|\mathbf{Q}_p| \le m$ and

$$\mathrm{Sub}_k(\mathcal{O}(\mathscr{G})|\mathbf{Q}_p) \cap \mathrm{prof}_k^{-1}(\Pi) = \mathcal{S}(\mathscr{G}, p),$$

where $\Pi = \mathrm{prof}(\mathcal{S}(\mathscr{G}, p))$. We call $\mathbf{Q}_p$ a *characterizing context set* of p.

Algorithm. Our dual learner turns out to be considerably simpler than its primal cousin. While the primal learner uses two profile sets, the dual learner assigns just a single profile to each nonterminal. This corresponds to the fact that the context-profiles play no role in constructing a structure-k-bounded grammar and that the (k, m)-FCP$_{\mathrm{prof}}$ is preserved under the normalization which converts a grammar into an equivalent one $\mathscr{G}'$ where $\mathrm{prof}_k(\mathcal{S}(\mathscr{G}', p))$ is a singleton for all abstract atomic types p of $\mathscr{G}'$, where it is not necessarily the case for the (k, m)-FKP$_{\mathrm{prof}}$.

Hereafter we fix a learning target $\mathbf{T}_* \subseteq \mathrm{LNF}^o_\varnothing(\varDelta)$ which is generated by $\mathscr{G}_* = (\varSigma, \varDelta, \mathcal{H}, \mathscr{I}) \in \mathbb{G}(\varOmega, n, k)$ with the (k, m)-FCP$_{\mathrm{prof}}$. We write $\mathbf{Q}^{[\pi]} = \mathrm{Sub}_k(\mathbf{T}_*|\mathbf{Q}) \cap \mathrm{prof}_k^{-1}(\pi)$ for a k-bounded profile π.

Our learner (Algorithm 2) constructs a context-k-bounded ACG $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{F}, \mathbf{B}, \mathbf{K}) = (\varGamma, \varDelta, \mathcal{J}, \mathscr{J})$ from three sets $\mathbf{F} \subseteq \mathrm{Con}^\varOmega_k(\mathbf{D})$, $\mathbf{B} \subseteq \mathrm{Glue}^{\varOmega,n}_k(\mathbf{D})$, and $\mathbf{K} \subseteq \mathrm{Sub}^\varOmega_k(\mathbf{D})$, where $\mathbf{D}$ is a finite set of positive examples.

Algorithm 2. Learning ACGs in $\mathbb{G}(\varOmega, n, k)$ with (k, m)-FCP$_{\mathrm{prof}}$

```
Data: A positive presentation $T_1, T_2, \dots$ of $\mathbf{T}_*$; membership oracle on $\mathbf{T}_*$;
Result: A sequence of ACGs $\mathscr{G}_1, \mathscr{G}_2, \dots$;
let $\mathbf{D} := \mathbf{F} := \mathbf{B} := \mathbf{K} := \varnothing$; $\hat{\mathscr{G}} := \mathcal{G}(\mathbf{F}, \mathbf{B}, \mathbf{K})$;
for $i = 1, 2, \dots$ do
  let $\mathbf{D} := \mathbf{D} \cup \{T_i\}$; $\mathbf{K} := \mathrm{Sub}^\varOmega_k(\mathbf{D})$;
  if $\mathbf{D} \nsubseteq \mathcal{O}(\hat{\mathscr{G}})$ then
    let $\mathbf{B} := \mathrm{Glue}^{\varOmega,n}_k(\mathbf{D})$;
    let $\mathbf{F} := \mathrm{Con}^\varOmega_k(\mathbf{D})$;
  end if
  output $\hat{\mathscr{G}} = \mathcal{G}(\mathbf{F}, \mathbf{B}, \mathbf{K})$ as $\mathscr{G}_i$;
end for
```

Each abstract atomic type of our grammar is a pair of a finite subset of $\mathbf{F} \cap \mathrm{AL}^\alpha_k(\varDelta)$ of cardinality at most m and a profile π whose type is α:

$$A_\varGamma = \{ [\![\mathbf{Q}, \pi]\!] \mid \pi \text{ is a } k\text{-bounded profile of type } \alpha \in \varOmega, \\ \mathbf{Q} \subseteq \mathbf{F} \cap \mathrm{AL}^{\alpha \to o}_k(\varDelta) \text{ and } 1 \le |\mathbf{Q}| \le m \} .$$

We have $|A_\varGamma| \le |\mathbf{F}|^m \ell$ for ℓ the number of possible profiles. We have only one distinguished type:

$$\mathscr{J} = \{ [\![\{\lambda y.y\},\, (\lambda z^o.z)]\!] \} .$$

We define $\mathcal{J}([\![\mathbf{Q}, \pi]\!])$ to be the type of π.

We have an abstract constant $c \in C_\varGamma$ such that

$$\tau_\varGamma(c) = [\![\mathbf{Q}_1, \pi_1]\!] \to \cdots \to [\![\mathbf{Q}_j, \pi_j]\!] \to [\![\mathbf{Q}_0, \pi_0]\!] \text{ with } j \le n\,, \quad \mathcal{J}(c) = P \in \mathbf{B}\,,$$

if

– $\pi_0 = P\pi_1 \dots \pi_j$,

– $|Q(PS_1 \dots S_j)|_\beta \in \mathbf{T}_*$ for all $Q \in \mathbf{Q}_0$ and $S_i \in \mathbf{Q}_i^{[\pi_i]} \cap \mathbf{K}$.

The second clause is checked with the aid of the membership oracle. By the construction, $\mathrm{prof}(|\mathscr{J}(M)|_\beta) \in \pi$ for every $M \in \mathrm{LNF}_{\varnothing}^{[\![\mathbf{Q},\pi]\!]}(\Gamma)$. Thus the grammar $\hat{\mathscr{G}}$ is substructure-k-bounded.

Lemma 13.
If $\mathbf{F} \subseteq \mathbf{F}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F}',\mathbf{B},\mathbf{K}))$.
If $\mathbf{B} \subseteq \mathbf{B}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B}',\mathbf{K}))$.
If $\mathbf{K} \subseteq \mathbf{K}'$, *then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \supseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K}'))$.

Lemma 14. *Let* $\mathbf{Q}_p$ *be a characterizing set of each atomic type* $p \in A_\Sigma$ *of the target grammar* $\mathscr{G}_*$. *Then* $\mathbf{Q}_p \subseteq \mathrm{Con}_k^\Omega(\mathbf{T}_*)$. *Moreover, if* $\mathbf{Q}_p \subseteq \mathbf{F}$ *for all* $p \in A_\Sigma$ *and* $\mathcal{H}(c) \in \mathbf{B}$ *for all* $c \in C_\Sigma$, *then* $\mathbf{T}_* \subseteq \mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K}))$ *for any* $\mathbf{K}$.

We say that an abstract constant c of type $[\![\mathbf{Q}_1,\pi_1]\!] \to \cdots \to [\![\mathbf{Q}_j,\pi_j]\!] \to [\![\mathbf{Q}_0,\pi_0]\!]$ is *invalid* if $|Q(\mathscr{J}(c)S_1 \dots S_j)|_\beta \notin \mathbf{T}_*$ for some $S_i \in \mathbf{Q}_i^{[\pi_i]}$ and $Q \in \mathbf{Q}_0$.

Lemma 15. *For every* $\mathbf{F}$ *and* $\mathbf{B}$, *there is a finite set* $\mathbf{K} \subseteq \mathrm{Sub}_k^\Omega(\mathbf{T}_*)$ *of cardinality* $n|\mathbf{B}||A_\Gamma|^{n+1}$ *such that* $\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$ *has no invalid constant.*

Lemma 16. *If* $\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})$ *has no invalid constant, then* $\mathcal{O}(\mathcal{G}(\mathbf{F},\mathbf{B},\mathbf{K})) \subseteq \mathbf{T}_*$.

Theorem 2. *Algorithm 2 successfully learns all grammars in* $\mathbb{G}(\Omega,n,k)$ *with the* (k,m)-$\mathrm{FCP}_{\mathrm{prof}}$.

A remark similar to the one on the efficiency of Algorithm 1 applies to Algorithm 2.

Acknowledgement. This work was supported in part by MEXT/JSPS Kakenhi (24106010, 26330013) and NII joint research project “Algorithmic Learning of Nonlinear Formalisms Based on Distributional Learning”.

References

1. Bloem, R., Engelfriet, J.: A comparison of tree transductions defined by monadic second order logic and by attribute grammars. J. Comput. Syst. Sci. **61**, 1–50 (2000)
2. Böhm, C., Coppo, M., Dezani-Ciancaglini, M.: Termination tests inside λ-calculus. In: Salomaa, A., Steinby, M. (eds.) Automata, Languages and Programming. LNCS, vol. 52, pp. 95–110. Springer, Heidelberg (1977)
3. Clark, A.: Learning context free grammars with the syntactic concept lattice. In: Sempere and García [11], pp. 38–51
4. Clark, A., Yoshinaka, R.: Distributional learning of parallel multiple context-free grammars. Mach. Learn. **96**(1–2), 5–31 (2014). doi:10.1007/s10994-013-5403-2
5. Kanazawa, M.: Parsing and generation as datalog queries. In: Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, Prague, Czech Republic, pp. 176–183 (2007)

6. Kanazawa, M.: A lambda calculus characterization of MSO definable tree transductions (abstract). Bull. Symbolic Logic **15**(2), 250–251 (2009)
7. Kanazawa, M.: Parsing and generation as datalog query evaluation. IfColog J. Logics Their Appl. (to appear). http://research.nii.ac.jp/~kanazawa/publications/pagadqe.pdf
8. Kanazawa, M.: Almost affine lambda terms. In: Indrzejczak, A., Kaczmarek, J., Zawidzki, M. (eds.) Trends in Logic XIII, pp. 131–148. Łódź University Press, Łódź (2014)
9. Kasprzik, A., Yoshinaka, R.: Distributional learning of simple context-free tree grammars. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS, vol. 6925, pp. 398–412. Springer, Heidelberg (2011)
10. Salvati, S.: Encoding second order string ACG with deterministic tree walking transducers. In: Wintner, S. (ed.) Proceedings of FG 2006: The 11th conference on Formal Grammar. FG Online Proceedings, pp. 143–156. CSLI Publications, Stanford (2007)
11. Sempere, J.M., García, P. (eds.): Grammatical Inference: Theoretical Results and Applications. Proceedings of 10th International Colloquium, ICGI 2010, Valencia, Spain, 13–16 September 2010. LNCS. Springer, Heidelberg (2010)
12. Yoshinaka, R.: Polynomial-time identification of multiple context-free languages from positive data and membership queries. In: Sempere, J.M., García, P. (eds.) ICGI 2010. LNCS, vol. 6339, pp. 230–244. Springer, Heidelberg (2010)
13. Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In: Mauri, G., Leporati, A. (eds.) DLT 2011. LNCS, vol. 6795, pp. 429–440. Springer, Heidelberg (2011)
14. Yoshinaka, R., Kanazawa, M.: Distributional learning of abstract categorial grammars. In: Pogodalla, S., Prost, J.-P. (eds.) Logical Aspects of Computational Linguistics. LNCS, vol. 6736, pp. 251–266. Springer, Heidelberg (2011)