General Perspective on Distributionally Learnable Classes

Ryo Yoshinaka
Kyoto University, Japan
ry@i.kyoto-u.ac.jp

Abstract

Several algorithms have been proposed to learn different subclasses of context-free grammars based on the idea generically called *distributional learning*. Those techniques have been applied to many formalisms richer than context-free grammars like multiple context-free grammars, simple context-free tree grammars and others. The learning algorithms for those different formalisms are actually quite similar to each other. We in this paper give a uniform view on those algorithms.

1 Introduction

Approaches based on the idea generically called *distributional learning* have been making great success in the algorithmic learning of various subclasses of context-free grammars (CFGs) (Clark, 2010c; Yoshinaka, 2012). Those techniques are applied to richer formalisms as well. The formalisms studied so far include multiple CFGs (Yoshinaka, 2011a), simple context-free tree grammars (CFTGs) (Kasprzik and Yoshinaka, 2011), second-order abstract categorial grammars (Yoshinaka and Kanazawa, 2011), parallel multiple CFGs (Clark and Yoshinaka, 2014), conjunctive grammars (Yoshinaka, 2015) and others. The goal of this paper is to present a uniform view on those algorithms.

Every grammar formalism for which distributional learning techniques have been proposed so far generate their languages through context-free derivation trees, whose nodes are labeled by production rules. The formalism and grammar rules determine how a context-free derivation tree τ is mapped to a derived object $\tilde{\tau} = d$. A context-free derivation tree τ can be decomposed into a subtree σ and a tree-context χ so that $\tau = \chi[\sigma]$. The subtree determines a substructure $s = \tilde{\sigma}$ of d and the tree-context determines a contextual structure $c = \tilde{\chi}$ in which the substructure is plugged to form the derived object $d = c \odot s$, where we represent the plugging operation by $\odot$. In the CFG case, c is a string pair $\langle l, r\rangle$ and s is a string u and $\langle l, r\rangle \odot u = lur$, which may correspond to a derivation $I \overset{*}{\Rightarrow} lXr \overset{*}{\Rightarrow} lur$ where I is the initial symbol and X is a nonterminal symbol. In richer formalisms those substructures and contexts may have richer structures, like tuples of strings or λ-terms. A learner does not know how a given example d is derived by a hidden grammar behind the observed examples. A learner based on distributional learning simply tries all the possible decompositions of a positive example into arbitrary two parts c' and s' such that $d = c' \odot s'$ where some grammar may derive d thorough a derivation tree $\tau' = \chi'[\sigma']$ with $\tilde{\chi}' = c'$ and $\tilde{\sigma}' = s'$. Based on observation on the relation between substructures and contexts collected from given examples, a hypothesis grammar is computed. We call properties on grammars with which distributional learning approaches work *distributional properties*.

This paper first formally defines grammar formalisms based on context-free derivation trees. We then show that grammars with different distributional properties are learnable by standard distributional learning techniques if the formalism satisfies some conditions, which include polynomial-time decomposability of objects into contexts and

Proceedings of the 14th Meeting on the Mathematics of Language (MoL 14), pages 87–98,
Chicago, USA, July 25–26, 2015.

substructures. In addition, we discuss cases where we cannot enumerate all of the possible contexts and substructures.

2 Σ-grammars

There is a number of ways to represent a *language*, a subset of an object set $\mathbb{O}_*$, whose elements are typically strings, trees but anythings encodable are eligible. Formalisms this paper discusses generate objects in $\mathbb{O}_*$ through context-free derivation trees τ, which are mapped to an element $d \in \mathbb{O}_*$ in a uniform way. The map is inductively defined and computed. Each derivation subtree τ' of τ also determines an object, which we call a *substructure* of d. Each substructure is not necessarily a member of $\mathbb{O}_*$. For example, nonterminal symbols of multiple CFGs (Seki et al., 1991) derive n-tuples of strings, where the value n is unique to each nonterminal, while the languages generated by multiple CFGs are still simply string sets. A generalization of the CFG formalism is specified by kinds of objects that each nonterminal generates and admissible operations over those objects.

Let $\mathbb{O}$ be a set of *objects*, which are identified with their codes of finite length. We have a set Ω of finite representations O which are interpreted as subsets $\mathbb{O}_O$ of $\mathbb{O}$ through an effective procedure. By a *sort* we flexibly refer to $O \in \Omega$ or $\mathbb{O}_O \subseteq \mathbb{O}$. We also have an indexed family of computable functions from tuples of objects of some sorts to objects of some sort. Let $\mathbb{F}$ be a set of function names or function indices f, which represent functions $\tilde{f}$. By $\mathbb{O}_1 \times \cdots \times \mathbb{O}_n \to \mathbb{O}_0$ we denote the set of functions whose domain is $\mathbb{O}_1 \times \cdots \times \mathbb{O}_n$ and codomain is $\mathbb{O}_0$. By $\mathbb{F}_{O_0,O_1,\ldots,O_n}$, we denote the set of function names $f \in \mathbb{F}$ with $\tilde{f} \in \mathbb{O}_{O_1} \times \cdots \times \mathbb{O}_{O_n} \to \mathbb{O}_{O_0}$. We assume that the domain sorts $O_1, \ldots, O_n$ and the codomain sort O_0 are easily computed from f. We specify a class of grammars by a triple, which we call a *signature*, $\Sigma = \langle \Omega, \mathbb{F}, O_* \rangle$ where $O_* \in \Omega$ is a special sort of objects. We write $\mathbb{O}_*$ for $\mathbb{O}_{O_*}$.

A *context-free* Σ*-grammar* (Σ*-grammar* for short) is a tuple $G = \langle N, \sigma, F, P, I \rangle$ where N is a finite set of *nonterminal symbols*, $I \subseteq N$ is a set of *initial symbols*, $\sigma \in N \to \Omega$ is a *sort assignment* on nonterminals such that $\sigma(X) = O_*$ for all $X \in I$, $F \subseteq \mathbb{F}$ is a finite set of function names, and P is a finite set of *production rules*, which are elements of $N \times F \times N^*$. Each production rule is denoted as

$$X_0 \leftarrow f\langle X_1, \ldots, X_n \rangle$$

where $X_0, \ldots, X_n \in N$ and $f \in \mathbb{F}_{O_0,O_1,\ldots,O_n}$ for $\sigma(X_i) = O_i$. For each $O \in \Omega$, $N_O = \sigma^{-1}(O) \subseteq N$ is the set of *O-nonterminals* which are assigned the sort O. By $\mathcal{G}(\Sigma)$ we denote the class of Σ-grammars.

A Σ-grammar defines its language via derivation trees, which are recursively defined as follows.

- If τ_i are X_i-derivation trees for $i = 1, \ldots, n$ and ρ is a rule of the form $X_0 \leftarrow f\langle X_1, \ldots, X_n \rangle$, then the term $\tau_0 = \rho[\tau_1, \ldots, \tau_n]$ is an X_0-derivation tree. Its *yield* $\tilde{\tau}_0$ is $\tilde{f}(\tilde{\tau}_1, \ldots, \tilde{\tau}_n) \in \mathbb{O}_{\sigma(X_0)}$ where $\tilde{\tau}_i$ is the yield of τ_i.

The case where $n = 0$ gives the base of this recursive definition. An X-derivation tree is *complete* if $X \in I$. The yield of any X-derivation tree is called an X*-substructure*. By $\mathcal{S}(G, X)$ we denote the set of X-substructures. The language of G is $\mathcal{L}(G) = \bigcup_{X \in I} \mathcal{S}(G, X)$, which we call a Σ*-language*. In other words, $\mathcal{L}(G)$ is the set of the yields of complete derivation trees. The class of Σ-languages is denoted by $\mathcal{L}(\Sigma)$.

Distributional learning is concerned with what X*-derivation contexts* represent. An X-derivation context is obtained by replacing an occurrence of an X-derivation tree in a complete derivation tree by a special symbol $\Box_{\sigma(X)}$. Accordingly the yield $\tilde{\chi}$ of an X-derivation context χ should be a finite representation of a function that gives $\widetilde{\chi[\tau]}$ when applied to $\tilde{\tau}$ for any X-derivation tree τ. We assume to have a set $\mathbb{E}_O$ of representations of functions from $\mathbb{O}_O$ to $\mathbb{O}_*$ for $O \in \Omega$ to which the yields of derivation contexts belong.

- $\Box_X$ is an X-derivation context for all $X \in I$ and its yield $\Box_{O_*} \in \mathbb{E}_{O_*}$ represents the identity function on $\mathbb{O}_*$,
- For an X-derivation context χ_0, a rule $\rho = X \leftarrow f\langle X_1, \ldots, X_n \rangle$ and X_i-derivation trees τ_i for $i \in \{1, \ldots, n\} - \{j\}$, the term χ obtained by replacing $\Box_X$ in χ_0 by $\rho[\tau_1, \ldots, \tau_{j-1}, \Box_{X_j}, \tau_{j+1}, \ldots, \tau_n]$ is an X_j-derivation context. Its yield $\tilde{\chi} \in \mathbb{E}_{\sigma(X_j)}$, which

is denoted as

$$\tilde{\chi} = \tilde{\chi_0} \odot \tilde{f}(\tilde{\tau}_1, \ldots, \tilde{\tau}_{j-1}, \Box_{\sigma(X_j)}, \tilde{\tau}_{j-1}, \ldots, \tilde{\tau}_n),$$

represents the function $\phi \in \mathbb{O}_{\sigma(X_j)} \to \mathbb{O}_*$ such that for all $s \in \mathbb{O}_{\sigma(X_j)}$,

$$\phi(s) = \phi_0(\tilde{f}(\tilde{\tau}_1, \ldots, \tilde{\tau}_{j-1}, s, \tilde{\tau}_{j+1}, \ldots, \tilde{\tau}_n)),$$

where ϕ_0 is the function represented by $\tilde{\chi}_0$.

The yield of any X-derivation context is called an *X-context*. By $\mathcal{C}(G, X)$ we denote the set of X-contexts. For $c \in \mathcal{C}(G, X)$ and $s \in \mathcal{S}(G, X)$, $c \odot s$ is the result of the application of the function represented by c to s.

3 Context-substructure relation

By $\mathbb{S}$ and $\mathbb{C}$ we denote the set of substructures and contexts, respectively, which can be obtained by some grammar in $\mathcal{G}(\Sigma)$:

$$\mathbb{S} = \bigcup_{O \in \Omega} \mathbb{S}_O \text{ and } \mathbb{C} = \bigcup_{O \in \Omega} \mathbb{C}_O \text{ where}$$
$$\mathbb{S}_O = \bigcup \{\, \mathcal{S}(G, X) \mid X \text{ is an } O\text{-nonterminal of some } G \in \mathcal{G}(\Sigma) \,\}$$
$$\mathbb{C}_O = \bigcup \{\, \mathcal{C}(G, X) \mid X \text{ is an } O\text{-nonterminal of some } G \in \mathcal{G}(\Sigma) \,\}.$$

We write $\mathbb{S}_*$ for $\mathbb{S}_{O_*}$. Note that the above definition is relative to Σ. Even if $\mathbb{O}_{O_1} = \mathbb{O}_{O_2}$ for different $O_1, O_2 \in \Omega$, it can be the case that $\mathbb{S}_{O_1} \neq \mathbb{S}_{O_2}$ and $\mathbb{C}_{O_1} \neq \mathbb{C}_{O_2}$. Though usually $\mathbb{O}_O$ has a definition independent from Σ, it is possible to specify $\mathbb{O}_O$ in terms of the signature so that $\mathbb{S}_O = \mathbb{O}_O$. Clearly if $s \in \mathbb{S}_O$ and $c \in \mathbb{C}_O$, then there is a grammar $G \in \mathcal{G}(\Sigma)$ generating $c \odot s$ using a nonterminal of sort O. Therefore, $c \odot s$ is well defined for any $s \in \mathbb{S}_O$ and $c \in \mathbb{C}_O$ without specifying a particular Σ-grammar. Similarly $c \odot \tilde{f}(s_1, \ldots, s_{j-1}, \Box_{O_j}, s_{j+1}, s_n)$ is well defined for any $c \in \mathbb{C}_{O_0}$, $f \in \mathbb{F}_{O_0, \ldots, O_n}$ and $s_i \in \mathbb{S}_{O_i}$. This operation is generalized to sets $S \subseteq \mathbb{S}_O$ and $C \subseteq \mathbb{C}_O$ in the straightforward way, like $C \odot S = \{\, c \odot s \mid c \in C \text{ and } s \in S \,\}$.

Hereafter, whenever we write $c \odot s$ and $\tilde{f}(s_1, \ldots, s_n)$, we assume they are well-formed. That is, the domains of the functions represented by c and f match the sorts to which s and $s_1, \ldots, s_n$ belong, respectively. Accordingly we drop the subscript O from $\Box_O$ and write $\tilde{f}(s_1, \ldots, s_{j-1}, \Box, s_{j+1}, \ldots, s_n)$. When we have a substructure set S, we assume $S \subseteq \mathbb{S}_O$ for some $O \in \Omega$. We often identify s with $\{s\}$ unless confusion arises. Also we assume $\mathbb{S}_O \neq \varnothing$ for all $O \in \Omega$. The same assumptions apply to contexts.

We are interested in whether the composition $c \odot s$ belongs to a concerned language $L \in \mathcal{L}(\Sigma)$. Clark (2010b) has introduced *syntactic concept lattices* to analyze the context-substring relation on string languages and particularly to design a distributional learning algorithm for CFLs. Generalizing his discussion, we define an O-concept lattice $\mathfrak{B}_O(L)$ of a language $L \subseteq O_*$ for respective sorts $O \in \Omega$. Assuming L and O understood from the context, let us write

$$S^\ddagger = \{\, c \in \mathbb{C}_O \mid c \odot S \subseteq L \,\},$$
$$C^\dagger = \{\, s \in \mathbb{S}_O \mid C \odot s \subseteq L \,\}$$

for $S \subseteq \mathbb{S}_O$ and $C \subseteq \mathbb{C}_O$. We write $S^\dagger$ for $S^{\ddagger\dagger}$ and $C^\ddagger$ for $C^{\dagger\ddagger}$.

We call a pair $\langle S, C \rangle \subseteq \mathbb{S}_O \times \mathbb{C}_O$ a *concept* iff $S^\dagger = C$ and $C^\ddagger = S$. For any $S \subseteq \mathbb{S}_O$ and $C \subseteq \mathbb{C}_O$, $\langle S^\dagger, S^\ddagger \rangle$ and $\langle C^\dagger, C^\ddagger \rangle$ are concepts. We call them the concepts *induced by* S and C, respectively. For two concepts $\langle S_1, C_1 \rangle$ and $\langle S_2, C_2 \rangle$ in $\mathfrak{B}_O(L)$, we write $\langle S_1, C_1 \rangle \leq_L^O \langle S_2, C_2 \rangle$ if $S_1 \subseteq S_2$, which is equivalent to $C_2 \subseteq C_1$. With this partial order, $\mathfrak{B}_O(L)$ is a complete lattice.

We can introduce a partial order to substructure sets based on the concepts that they induce. Let us write $S_1 \leq_L^O S_2$ if $S_2^\ddagger \subseteq S_1^\ddagger$. The relation represents the substitutability of S_1 for S_2.

Lemma 1. *The following three are equivalent for $S, T \subseteq \mathbb{S}_O$:*

- $S \leq_L^O T$,
- *$c \odot T \subseteq L$ implies $c \odot S \subseteq L$ for all $c \in \mathbb{C}_O$,*
- $T^\ddagger \odot S \subseteq L$.

If $S_i \leq_L^{O_i} T_i$ for $i = 1, \ldots, n$, then for any $f \in \mathbb{F}_{O_0, O_1, \ldots, O_n}$, we have $f(S_1, \ldots, S_n) \leq_L^{O_0} f(T_1, \ldots, T_n)$.

If $S_1 \leq_L S_2$ and $S_2 \leq_L S_1$, we write $S_1 \equiv_L S_2$.

4 Conditions to be distributionally learnable

Distributional learning algorithms decompose examples $d \in \mathbb{S}_*$ into contexts $c \in \mathbb{C}_O$ and substructures $s \in \mathbb{S}_O$ so that $c \odot s = d$. Then a *primal* approach uses substructures or sets of substructures as nonterminals of a conjecture grammar. We want each nonterminal $[\![S]\!]$ indexed by $S \subseteq \mathbb{S}_O$ to satisfy $\mathcal{S}(G, [\![S]\!]) = S^\dagger$. On the other hand, a *dual* approach uses contexts or sets of contexts as nonterminals where the semantics of the nonterminal is $\mathcal{S}(G, [\![C]\!]) = C^\dagger$. For an object $d \in \mathbb{S}_*$, $O \in \Omega$ and $\vec{O} = (O_0, \ldots, O_n)$ with $O_i \in \Omega$, we define

$$\mathbb{S}_{O|d} = \{\, s \in \mathbb{S}_O \mid c \odot s = d \text{ for some } c \in \mathbb{C}_O \,\}$$
$$\mathbb{C}_{O|d} = \{\, c \in \mathbb{C}_O \mid c \odot s = d \text{ for some } s \in \mathbb{S}_O \,\}$$
$$\mathbb{F}_{\vec{O}|d} = \{\, f \in \mathbb{F}_{\vec{O}} \mid c \odot f(s_1, \ldots, s_m) = d \text{ for some } c \in \mathbb{C}_{O_0|d} \text{ and } s_i \in \mathbb{S}_{O_i|d} \,\},$$

$\mathbb{S}_{|D} = \bigcup_{O \in \Omega}^{d \in D} \mathbb{S}_{O|d}$, $\mathbb{C}_{O|D} = \bigcup_{O \in \Omega}^{d \in D} \mathbb{C}_{O|d}$ and $\mathbb{F}_{|D} = \bigcup_{\vec{O} \in \Omega^*}^{d \in D} \mathbb{F}_{\vec{O}|d}$ for $D \subseteq O_*$. Let $\Omega_{|D} = \bigcup_{d \in D} \Omega_{|d}$ for $\Omega_{|d} = \{\, O \mid \mathbb{S}_{O|d} \neq \varnothing \,\}$.

We require $\mathcal{G}(\Sigma)$ to be a tractable formalism such that composition and decomposition can be done efficiently.

Assumption 1. *There are polynomial-time algorithms which*

- *decide whether $s \in \mathbb{S}_O$ from $s \in \mathbb{S}$ and $O \in \Omega$,*
- *compute $\tilde{f}(s_1, \ldots, s_n)$ from $s_i \in \mathbb{S}_{O_i}$ and $f \in \mathbb{F}_{O_0, O_1, \ldots, O_n}$,*
- *decide whether $c \in \mathbb{C}_O$ from $c \in \mathbb{C}$ and $O \in \Omega$,*
- *compute $c \odot s$ from $c \in \mathbb{C}_O$ and $s \in \mathbb{S}_O$ for any $O \in \Omega$.*
- *decide whether $s \in \mathcal{L}(G)$ from $s \in \mathbb{S}_*$ and $G \in \mathcal{G}(\Sigma)$,*

Assumption 2. *There is $p \in \mathbb{N}$ such that the arity of every $f \in \mathbb{F}$ is at most p.*

Assumption 3. *There are polynomial-time algorithms that compute* $\textsc{Sub}(d)$, $\textsc{Con}(d)$ *and* $\textsc{Fun}(d)$ *from $d \in \mathbb{S}_*$ such that $\mathbb{S}_{|d} \subseteq \textsc{Sub}(d) \subseteq \mathbb{S}$, $\mathbb{C}_{|d} \subseteq \textsc{Con}(d) \subseteq \mathbb{C}$ and $\mathbb{F}_{|d} \subseteq \textsc{Fun}(d) \subseteq \mathbb{F}$.*

Actually by Assumptions 2 and 3, one can derive the polynomial-time uniform membership decidability. Moreover, it is easy to filter out nonmembers of $\mathbb{S}_{|d}$, $\mathbb{C}_{|d}$ and $\mathbb{F}_{|d}$ from $\textsc{Sub}(d)$, $\textsc{Con}(d)$ and $\textsc{Fun}(d)$, respectively, but it is not necessary. Assumption 3 implies $|\Omega_{|d}|$ is polynomially bounded, since $O \in \Omega_{|d}$ iff $\mathbb{F}_{O, O_1, \ldots, O_n} \neq \varnothing$ for some $O_1, \ldots, O_n$.

We write $\textsc{Sub}_O(D) = \textsc{Sub}(D) \cap \mathbb{S}_O$, $\textsc{Con}_O(D) = \textsc{Con}(D) \cap \mathbb{C}_O$ and $\textsc{Fun}_{\vec{O}}(D) = \textsc{Fun}(D) \cap \mathbb{F}_{\vec{O}}$.

It is often the case that elements of Ω represents pairwise disjoint sets. Actually for any signature Σ, one can find $\Sigma' = \langle \Omega', \mathbb{F}', O_* \rangle$ that satisfies this condition such that $\mathcal{L}(\Sigma) = \mathcal{L}(\Sigma')$. Let $\Omega' = \{\, O' \mid O \in \Omega \,\} \cup \{O_*\}$ and $\mathbb{O}_{O'} = \mathbb{O}_O \times \{O\}$ for each $O \in \Omega - \{O_*\}$. For $f \in \mathbb{F}_{O_0, \ldots, O_n}$ with $\tilde{f}(s_1, \ldots, s_n) = s_0$, we have $f' \in \mathbb{F}'_{O'_0, \ldots, O'_n}$ with $\tilde{f}'(s'_1, \ldots, s'_n) = s'_0$ where $s'_i = (s_i, O_i)$ if $O_i \in \Omega - \{O_*\}$ and $s'_i = s_i$ if $O_i = O_*$. Clearly every Σ-grammar has an equivalent Σ'-grammar. Moreover, this makes it clear that from $s \in \mathbb{S}$ one can immediately specify the unique sort $O' \in \Omega'$ such that $s \in \mathbb{O}_{O'}$. Similarly we may assume that each $c \in \mathbb{C}$ has unique $O \in \Omega$ such that $c \in \mathbb{C}_O$ and finding that O is a trivial task. Hereafter we work under this assumption. By O and f we mean $\mathbb{O}_O$ and $\tilde{f}$ for notational convenience.

Example 1. A right regular grammar over an alphabet Δ is a Σ_{reg}-grammar for $\Sigma_{\mathrm{reg}} = \langle \{\Delta^*\}, \mathbb{F}, \Delta^* \rangle$. $\mathbb{F}$ has nullary functions which are members of $\Delta \cup \{\varepsilon\}$ and unary functions f_a for $f_a(w) = aw$ for all $w \in \Sigma^*$ for some $a \in \Delta$. Clearly the class of right regular grammars satisfies Assumptions 1, 2 and 3.

Example 2. A CFG is a Σ_{cfg}-grammar for $\Sigma_{\mathrm{cfg}} = \langle \{\Delta^*\}, \mathbb{F}, \Delta^* \rangle$ where each $f \in \mathbb{F}$ is represented as an $(n+1)$-dimension vector of strings $\langle u_0, \ldots, u_n \rangle$ such that $f(v_1, \ldots, v_n) = u_0 v_1 u_1 \ldots v_n u_n$ for all $v_i \in \Delta^*$. The class of CFGs itself satisfies Assumption 1 but not Assumptions 2 and 3, since we have no limit on n. But several normal forms fulfill Assumptions 2 and 3.

Example 3. Let $\Omega = \{\, O_1, O_2, \ldots \}$ where O_m denotes the set of m-tuples of strings. Linear context-free rewriting systems, equivalent to nondeleting multiple CFGs, are Σ_{mcfg}-grammars where $O_* = O_1 = \Delta^*$ and every $f \in \mathbb{F}_{O_{m_0}, O_{m_1}, \ldots, O_{m_n}}$ concatenates strings $u_{i,j}$ occurring in an input $\langle \langle u_{1,1}, \ldots, u_{1,m_1} \rangle, \ldots, \langle u_{n,1}, \ldots, u_{n,m_n} \rangle \rangle$ in some way to form an m_0-tuple of strings. The uniform membership problem of this class is PSPACE-

complete (Kaji et al., 1992). There are infinitely many ways to decompose a string d into substructures and contexts as $O_m \in \Omega_{|d}$ for all m. Assumptions 1 and 3 will be fulfilled when we restrict admissible functions so that $\mathbb{F}_{O_{m_0},\ldots,O_{m_n}} \neq \varnothing$ only if $n \leq p$ and $m_i \leq q$ for all i.

As is the case for multiple CFGs, Assumption 2 is often needed to make the uniform membership problem solvable in polynomial-time (Assumption 1).

5 Learning models

Learning algorithms in this paper work under three different learning models.

A *positive presentation* (*text*) of a language $L_* \subseteq \mathbb{O}_*$ is an infinite sequence $d_1, d_2, \cdots \in \mathbb{O}_*$ such that $L_* = \{\, d_i \mid i \geq 1 \,\}$. In the framework of *identification in the limit from positive data*, 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 d_i is given, it outputs a grammar G_i computed from $d_1, \ldots, d_i$. We say that a learning algorithm $\mathcal{A}$ *identifies G_* in the limit from positive data* if for any positive presentation $d_1, d_2, \ldots$ of $\mathcal{L}(G_*)$, there is an integer n such that $G_n = G_m$ for all $m \geq n$ and $\mathcal{L}(G_n) = \mathcal{L}(G_*)$. We say that $\mathcal{A}$ *identifies a class $\mathbb{G}$ of grammars in the limit from positive data* iff $\mathcal{A}$ identifies all $G \in \mathbb{G}$ in the limit from positive data.

We say that $\mathcal{A}$ *identifies a class $\mathbb{G}$ of grammars in the limit from positive data and membership queries* when we allow $\mathcal{A}$ to ask *membership queries* (MQs) to an oracle when it computes a hypothesis grammar. An instance of an MQ is an object $d \in \mathbb{O}_*$ and the oracle answers whether $d \in L_*$ in constant time.

The third model is the learning with a *minimally adequate teacher* (MAT). A learner is not given a positive presentation but it may ask *equivalence queries* (EQs) to an oracle in addition to MQs. An instance of an EQ is a grammar G. If $\mathcal{L}(G) = L_*$, the oracle answers "*Congratulations!*" and the learning process ends. Otherwise, the oracle returns a counterexample $d \in (L_* - \mathcal{L}(G)) \cup (\mathcal{L}(G) - L_*)$, which is called *positive* if $d \in L_* - \mathcal{L}(G)$ and *negative* if $d \in \mathcal{L}(G) - L_*$.

When we have an oracle, the learning task itself is trivial unless we show some favorable property on the learning efficiency.

6 Learnable subclasses

This section presents how Σ-grammars with distributional properties can be learned. Note that all of those properties are relative to Σ. We assume Σ-grammars $G_* = \langle N_*, \sigma_*, F_*, P_*, I_* \rangle$ in this section have no useless nonterminals or functions. That is, $\mathcal{S}(G_*, X) \neq \varnothing, \mathcal{C}(G_*, X) \neq \varnothing$ for all $X \in N_*$ and every $f \in F_*$ appears in some rule in P_*.

6.1 Substitutable Languages

Definition 1 (Clark and Eyraud (2007))**.** A language $L \in \mathcal{L}(\Sigma)$ is said to be *substitutable* if for any $O \in \Omega$, $c_1, c_2 \in \mathbb{C}_O$ and $s_1, s_2 \in \mathbb{S}_O$,

$$c_1 \odot s_1,\ c_1 \odot s_2,\ c_2 \odot s_1 \in L \text{ implies } c_2 \odot s_2 \in L\,.$$

The definition can be rephrased as follows:

$$s_1^{\ddagger} \cap s_2^{\ddagger} \neq \varnothing \text{ implies } s_1 \equiv_L s_2\,.$$

Example 4. Yoshinaka (2008) has proposed a learning algorithm for k, l-substitutable CFLs, which satisfy the following property:

$$x_1uy_1vz_1, x_1uy_2vz_1, x_2uy_1vz_2 \in L \implies x_2uy_2vz_2 \in L$$

for any $x_i, y_i, z_i \in \Delta^*$, $u \in \Delta^k$ and $v \in \Delta^l$. We define a signature $\Sigma_{k,l} = \langle \Omega_{k,l}, \mathbb{F}_{k,l}, O_* \rangle$ as follows. Let $\Omega_{k,l} = \{O_*\} \cup \{\, O_{u,v} \mid u \in \Delta^k \text{ and } v \in \Delta^l \,\} \cup \{\, O_u \mid u \in \Delta^{<k+l} \,\}$, where $O_* = \Delta^*$, $O_{u,v} = \{\, \overline{uwv} \mid w \in \Delta^* \,\}$ and $O_u = \{\overline{u}\}$. Here we put overlines to make elements of Ω pairwise disjoint. Let $\mathbb{F}_{k,l} = \{\, +_{\alpha,\beta} \mid \alpha, \beta \in \Omega - \{\Delta^*\} \,\} \cup \{\, \square_\alpha^{O_*} \mid \alpha \in \Omega - \{O_*\} \,\} \cup \Delta$. The binary function $+_{\alpha,\beta}$ concatenates two strings from sorts α and β and gives the right sort in $\Omega - \{O_*\}$. For example, $+_{\alpha,\beta} \in \mathbb{F}$ with $\alpha = O_{u,v}$, $\beta = O_w$ has codomain $O_{u,x}$ where x is the suffix of vw of length l. The unary operation $\square_\alpha^{O_*} \in \mathbb{F}_{O_*,\alpha}$ simply removes the overline and "promotes" $\bar{u} \in \alpha$ to $u \in O_*$. Δ consists of the nullary functions giving a single letter from $\overline{\Delta}$. It is not hard to see that every CFG has an equivalent $\Sigma_{k,l}$-grammar. Note that O_*-nonterminals never occur on the right hand side of a rule in a $\Sigma_{k,l}$-grammar. Hence $\mathbb{C}_{O_*}$ is just the singleton $\{\square_{O_*}\}$ such that $\square_{O_*} \odot u = u$ for all $u \in \Delta^*$, whereas $\mathbb{C}_\alpha \neq \mathbb{C}_{O_*}$ contains arbitrary pairs of strings $\langle l, r \rangle \in \Delta^* \times \Delta^*$ such that $\langle l, r \rangle \odot \bar{u} = lur$ for any $\bar{u} \in \alpha$. The $\Sigma_{k,l}$-substitutability is exactly the k, l-substitutability.

Theorem 1. *The class of substitutable Σ-languages is identifiable in the limit from positive data.*

The theorem follows Lemmas 2 and 3 below.

From a finite set D of positive examples, Algorithm 1 computes the grammar $\textsc{SubstP}(D) = \langle N, \sigma, F, P, I\rangle$ defined as follows:

- $N_O = \{\, [\![s]\!] \mid s \in \textsc{Sub}_O(D) \,\}$ for $O \in \Omega_{|D}$,
- $I = \{\, [\![s]\!] \mid s \in D \,\}$,
- $F = \textsc{Fun}(D)$,
- P consists of the rules of the form

$$[\![s_0]\!] \leftarrow f\langle [\![s_1]\!], \ldots, [\![s_n]\!]\rangle$$

where $f \in \textsc{Fun}_{O_0,\ldots,O_n}(D)$ for $[\![s_i]\!] \in N_{O_i}$ if there is $c \in \textsc{Con}_O(D)$ such that

$$c \odot s_0,\ c \odot f(s_1, \ldots, s_n) \in D\,.$$

Since we assume elements of Ω are pairwise disjoint, each $[\![s]\!]$ belongs to a unique sort. Otherwise, each nonterminal should be tagged with a sort like $[\![s, O]\!]$.

Algorithm 1 Learning substitutable Σ-grammars

Data: A positive presentation $d_1, d_2, \ldots$
Result: A sequence of grammars $G_1, G_2, \ldots$
let $\hat{G}$ be a grammar such that $\mathcal{L}(\hat{G}) = \varnothing$;
for $n = 1, 2, \ldots$ **do**
  let $D = \{d_1, \ldots, d_n\}$;
  if $D \nsubseteq \mathcal{L}(\hat{G})$ **then**
    let $\hat{G} = \textsc{SubstP}(D)$;
  end if
  output $\hat{G}$ as G_n;
end for

An alternative way to construct a grammar is to use contexts rather than substructures for nonterminals. One can replace $\textsc{SubstP}(D)$ in the algorithm by $\textsc{SubstD}(D)$ which is defined as follows.

- $N_O = \{\, [\![c]\!] \mid c \in \textsc{Con}_O(D) \,\}$ for $O \in \Omega_{|D}$,
- $I = \{\, [\![\Box_{O_*}]\!] \,\}$,
- $F = \textsc{Fun}(D)$,
- P consists of the rules of the form

$$[\![c_0]\!] \leftarrow f\langle [\![c_1]\!], \ldots, [\![c_n]\!]\rangle$$

where $f \in \textsc{Fun}_{O_0,\ldots,O_n}$ for $[\![c_i]\!] \in N_{O_i}$ if there are $s_i \in \textsc{Sub}_{O_i}(D)$ such that

$c_i \odot s_i \in D$ for all i and $c_0 \odot f(s_1, \ldots, s_n) \in D$.

The existing algorithms for different classes of substitutable languages (Clark and Eyraud, 2007; Yoshinaka, 2008; Yoshinaka, 2011a) are based on slight variants of SubstP. This paper shows the correctness of the algorithm using SubstD.

Lemma 2. *Let D be a finite subset of a Σ-substitutable language L_* and G the grammar output by* $\textsc{SubstD}(D)$. *Then $\mathcal{L}(G) \subseteq L_*$.*

Proof. One can show by induction on the derivation that if $s \in \mathcal{S}(G, [\![c]\!])$ then $c \odot s \in L_*$. Suppose that G has a rule $[\![c]\!] \leftarrow f\langle [\![c_1]\!], \ldots, [\![c_n]\!]\rangle$, $s_i \in \mathcal{S}(G, [\![c_i]\!])$ and $s = f(s_1, \ldots, s_n)$. The induction hypothesis says $c_i \odot s_i \in L_*$ for all i. By the rule construction, there are t_i for $i = 1, \ldots, n$ such that $c_i \odot t_i \in D \subseteq L_*$ and $c \odot f(t_1, \ldots, t_n) \in D \subseteq L_*$. We have $s_i \equiv_{L_*} t_i$ since they occur in the same context c_i. By Lemma 1, $c \odot f(s_1, \ldots, s_n) \in L_*$. □

Let $G_* = \langle N_*, \sigma_*, F_*, P_*, I_*\rangle$ be a Σ-grammar generating L_*. Fix $s_X \in \mathcal{S}(G_*, X)$ and $c_X \in \mathcal{C}(G_*, X)$ where $c_X = \Box_{O_*}$ for $X \in I_*$. Define D_* by

$$\begin{aligned} D_* = &\{\, c_X \odot s_X \mid X \in N_* \,\} \\ &\cup \{\, c_{X_0} \odot f(s_{X_1}, \ldots, s_{X_n}) \\ &\qquad \mid X_0 \leftarrow f\langle X_1, \ldots, X_n\rangle \in P_* \,\}\,. \end{aligned}$$

Lemma 3. *If $D_* \subseteq D$, then $\mathcal{S}(G_*, X) \subseteq \mathcal{S}(\textsc{SubstD}(D), [\![s_X]\!])$ for all X.*

Proof. Let $G = \textsc{SubstD}(D)$. If G_* has a rule $X_0 \leftarrow f\langle X_1, \ldots, X_n\rangle$ then G has the corresponding rule $[\![c_{X_0}]\!] \leftarrow f\langle [\![c_{X_1}]\!], \ldots, [\![c_{X_n}]\!]\rangle$, since

$$c_{X_0} \odot f(s_{X_1}, \ldots, s_{X_n}), c_{X_i} \odot s_{X_i} \in D\,.$$

In particular since c_X for $X \in I$ is the identity function $\Box_{O_*}$, the corresponding nonterminal $[\![c_X]\!] = [\![\Box_{O_*}]\!]$ is the initial symbol of G, too. □

This shows that we do not need too many data to achieve a right grammar, since $|D_*| \leq |P_*| + |N_*|$, where $|\cdot|$ denotes the cardinality of a set. Moreover, it is easy to see Algorithm 1 updates its conjecture in polynomial time in the total size of D by Assumptions 1, 2 and 3.

6.2 Finite kernel property

Definition 2 (Clark et al. (2009), Yoshinaka (2011b))**.** A nonempty finite set $S \subseteq \mathbb{S}_{\sigma(X)}$ is called a *k-kernel* of a nonterminal X if $|S| \leq k$ and

$$\mathcal{S}(G, X) \equiv_{\mathcal{L}(G)} S\,.$$

A Σ-grammar G is said to have the *k-finite kernel property* (k-FKP) if every nonterminal X has a k-kernel S_X.

Theorem 2. *Under Assumptions 1, 2 and 3, Algorithm 2 identifies Σ-grammars with the k-FKP in the limit from positive data and membership queries.*

Algorithm 2 Learning Σ-grammars with k-FKP

Data: A positive presentation $d_1, d_2, \dots$ of L_*;
Result: A sequence of Σ-grammars $G_1, G_2, \dots$;
let $D := K := F := J := \varnothing$;
let $\hat{G} := \textsc{Primal}_k(K, F, J)$;
for $n = 1, 2, \dots$ **do**
  let $D := D \cup \{d_n\}$; $J := \textsc{Con}(D)$;
  if $D \notin \mathcal{L}(\hat{G})$ **then**
    let $K := \textsc{Sub}(D)$ and $F := \textsc{Fun}(D)$;
  end if
  output $\hat{G} = \textsc{Primal}_k(K, F, J)$ as G_n;
end for

The conjecture grammar $\textsc{Primal}_k(K, F, J) = \langle N, \sigma, F, P, I\rangle$ of Algorithm 2 is defined from finite sets of substructures $K \subseteq \mathbb{S}$, functions $F \subseteq \mathbb{F}$ and contexts $J \subseteq \mathbb{C}$. The subsets of those sets corresponding to respective sorts are denoted as $K_O = K \cap \mathbb{S}_O$, $J_O = J \cap \mathbb{C}_O$ and $F_{\vec{O}} = F \cap \mathbb{F}_{\vec{O}}$.

- $N_O = \{\, [\![S]\!] \mid S \subseteq K_O \text{ with } 1 \leq |S| \leq k \,\}$ for each $O \in \Omega_{|D}$,
- $I = \{\, [\![S]\!] \in N_{O_*} \mid S \subseteq L \,\}$,
- P consists of the rules of the form

$$[\![S_0]\!] \leftarrow f\langle [\![S_1]\!], \dots, [\![S_n]\!]\rangle$$

where $f \in F_{O_0, O_1, \dots, O_n}$ for $[\![S_i]\!] \in N_{O_i}$ if

$$({S_0}^{\ddagger} \cap J_{O_0}) \odot f(S_1, \dots, S_n) \subseteq L_*\,. \quad (1)$$

The grammar is constructed by the aid of finitely many MQs. $\textsc{Primal}_k(K, F, J)$ can be computed in polynomial time by Assumptions 1, 2 and 3. A rule $[\![S_0]\!] \leftarrow f\langle [\![S_1]\!], \dots, [\![S_n]\!]\rangle$ is compatible with the semantics of the nonterminals if $S_0^{\dagger} \supseteq f(S_1^{\dagger}, \dots, S_n^{\dagger})$, which is equivalent to

$${S_0}^{\ddagger} \odot f(S_1, \dots, S_n) \subseteq L_* \quad (2)$$

by Lemma 1. However, this condition (2) cannot be checked by finitely many MQs. The condition (1) can be seen as an approximation of (2), which is decidable by finitely many MQs. Clearly (2) implies (1) but not vice versa. If a rule satisfies (1) but not (2), we call the rule *incorrect*. If a rule is incorrect, there is a *witness* $c \in \mathbb{C}_{O_0} - J_{O_0}$ such that $c \odot S_0 \in L_*$ and $c \odot f(S_1, \dots, S_n) \notin L_*$.

Lemma 4. *For every finite $K \subseteq \mathbb{S}$ and $F \subseteq \mathbb{F}$ there is $J \subseteq \mathbb{C}$ such that $\hat{G} = \textsc{Primal}(K, F, J)$ has no incorrect rules and $|J| \leq |F||K|^{k(p+1)}$, in which case $\mathcal{L}(\hat{G}) \subseteq L_*$.*

Let S_X be a k-kernel of each nonterminal X of a grammar $G_* = \langle N_*, \sigma_*, F_*, P_*, I_*\rangle$ generating L_*.

Lemma 5. *There is a finite subset $D \subseteq L_*$ such that $S_X \subseteq \mathbb{S}_{|D}$ for all $X \in N_*$, $F_* \subseteq \mathbb{F}_{|D}$ and $|D| \leq k|N_*| + |P_*|$. Moreover, if $S_X \subseteq K$ for all $X \in N_*$ and $F_* \subseteq F$, then $L_* \subseteq \mathcal{L}(\hat{G})$.*

We prove Theorem 2 discussing the efficiency.

Proof of Theorem 2. Clearly Algorithm 2 updates its conjecture in polynomial time in the data size. Polynomially (in the size of G_*) many positive examples will stabilize K and F by Lemma 5. After K and F stabilized, all the incorrect rules will be removed with at most polynomially (in $|K||F|$) many examples by Lemma 4. After that point Algorithm 2 never changes the conjecture, which generates the target language L_*. □

6.3 Congruential grammars

Definition 3 (Clark (2010a))**.** A Σ-grammar G is said to be *congruential* if every $s \in \mathcal{S}(G, X)$ is a 1-kernel of every $X \in N$.

Congruential Σ-grammars have the 1-FKP. Under the following additional assumption, this special case will be polynomial-time learnable with a minimally adequate teacher.

Assumption 4. *For any derivation tree τ, the size of its yield $\tilde{\tau}$ is polynomially bounded by that of τ.*

Theorem 3. *Under Assumptions 1, 2, 3 and 4, Algorithm 3 learns any language L_* generated by a*

congruential Σ-grammar G_* with a minimally adequate teacher in time polynomial in $|N_*|, |F_*|, \ell$ where ℓ is the total size of counterexamples given to the learner.

Algorithm 3 Learning congruential Σ-grammars

```
let K := F := J := ∅;
let Ĝ := PRIMAL_1(K, F, J);
for n = 1, 2, ... do
  if L(Ĝ) = L_* (equivalence query) then
    output Ĝ and halt;
  else if the given counterexample d is positive
  (d ∈ L_* − L(Ĝ)) then
    let K := K ∪ SUB(d) and F := F ∪ FUN(d);
  else
    let J := J ∪ WITNESSP(τ_d, □_{O_*}) where τ_d
    is an (implicit) parse tree of d by Ĝ
  end if
  let Ĝ = PRIMAL_1(K, F, J);
end for
```

Algorithm 3 uses the same grammar construction PRIMAL as Algorithm 2 where the parameters K and F are calculated from positive counterexamples given by the oracle. On the other hand, J is computed in a different way. By Lemma 4, when the oracle answers a negative counterexample d towards an EQ, our conjecture $\hat{G}$ must use an incorrect rule to derive d. To find and remove such an incorrect rule, Algorithm 3 calls a subroutine WITNESSP with input $(\tau_d, \Box)$, where τ_d is a derivation tree of $\hat{G}$ whose yield is d. To be precise, τ_d does not have to be a derivation tree. Rather what we require is that for each $s \in \mathbb{S}_{|d}$, one can compute at least one tuple of $s_1, \ldots, s_n \in \mathbb{S}_{|d}$ and $f \in \mathbb{F}_{|d}$ such that $s = f(s_1, \ldots, s_n)$ and the height of the lowest derivation tree of each s_i is strictly lower than that of s. Indeed one can do this in polynomial time by a dynamic programming method from $\textsc{Sub}(d)$ and $\textsc{Fun}(d)$. Yet for explanatory easiness, we treat such information as an (implicit) derivation tree τ_d. The procedure WITNESSP returns a context that witnesses an incorrect rule that contributes to generating d by searching τ_d recursively calling itself. The procedure WITNESSP in general takes a pair (τ, c) such that τ is an $[\![s]\!]$-derivation tree of $\hat{G}$ and $c \in s^\ddagger - \tilde{\tau}^\ddagger$. Let $\tau = \rho(\tau_1, \ldots, \tau_n)$ where $\rho = [\![s]\!] \leftarrow f\langle [\![s_1]\!], \ldots, [\![s_n]\!] \rangle$. If $c \odot f(s_1, \ldots, s_n) \notin L_*$ then the rule ρ is incorrect. So WITNESSP returns c which witnesses the incorrectness of the rule. Otherwise, we have

$$c \odot f(s_1, \ldots, s_n) \in L_*$$
$$c \odot f(\tilde{\tau}_1, \ldots, \tilde{\tau}_n) \notin L_*$$

for the yields $\tilde{\tau}_i$ of τ_i. One can find i such that

$$c \odot f(s_1, \ldots, s_{i-1}, s_i, \tilde{\tau}_{i+1}, \ldots, \tilde{\tau}_n) \in L_*$$
$$c \odot f(s_1, \ldots, s_{i-1}, \tilde{\tau}_i, \tilde{\tau}_{i+1}, \ldots, \tilde{\tau}_n) \notin L_* .$$

This means an incorrect rule is in τ_i. We call $\textsc{WitnessP}(\tau_i, c \odot f(s_1, \ldots, s_{i-1}, \Box, \tilde{\tau}_{i+1}, \ldots, \tilde{\tau}_n))$.

Lemma 6. *The procedure* $\textsc{WitnessP}(\tau_d, \Box)$ *runs in polynomial time in ℓ and $|d|$.*

Proof. The number of recursive calls of WITNESSP is no more than the height of τ_d, which is at most $|\mathbb{S}_{|d}|$. Let the instance of the j-th recursive call be (τ_j, c_j) and χ_j the derivation context for $c = \tilde{\chi_j}$. χ_{j+1} is obtained from χ_j by replacing at most p subtrees by a derivation tree whose yield is an element of K. By Assumption 4, the size of c_j and thus the size of an instance of an MQ is polynomially bounded by $|d|\ell$. WITNESSP runs in polynomial time. □

Lemma 7. *Each time Algorithm 3 receives a negative counterexample, at least one incorrect rule is removed.*

Lemma 8. *Let $G_* = \langle N_*, \sigma, F_*, P_*, I_* \rangle$ be a congruential grammar generating L_*. Each time Algorithm 3 receives a positive counterexample, the cardinality of the set $\{ X \in N_* \mid K \cap \mathcal{L}(G_*, X) = \varnothing \} \cup (F_* - F)$ decreases strictly.*

Proof of Theorem 3. Time between an EQ and another is polynomially bounded by Lemma 6. By Lemmas 5 and 8, Algorithm 3 gets at most $|N_*| + |F_*|$ positive counterexamples. The grammar $\hat{G} = \textsc{Primal}(K, F, J)$ is constructed from those positive counterexamples, so it has polynomially many rules. Therefore, by Lemma 7, after getting polynomially many negative counterexamples, which suppress all the incorrect rules, Algorithm 3 gets a right grammar representing L_*. □

6.4 Finite context property

Definition 4 (Clark (2010b), Yoshinaka (2011b)[1])**.** A nonempty finite set $C \subseteq \mathbb{C}$ is called a *k-context* of a nonterminal X if $|C| \leq k$ and

$$\mathcal{S}(G, X) \equiv_{\mathcal{L}(G)} C^\dagger .$$

A Σ-grammar G is said to have the *k-(weak) finite context property* (k-FCP) if every nonterminal X has a k-context C_X.

Theorem 4. *Under Assumptions 1, 2 and 3, Algorithm 4 identifies Σ-grammars with the k-FCP in the limit from positive data and membership queries.*

The theorem can be shown by an argument similar to the proof of Theorem 2 based on Lemmas 9 and 10 below. The discussion on the learning efficiency of Algorithm 2 is applied to Algorithm 4 as well.

Algorithm 4 Learning Σ-grammars with k-FCP

Data: A positive presentation $d_1, d_2, \dots$ of L_*;
Result: A sequence of Σ-grammars $G_1, G_2, \dots$;
let $D := J := F := K := \varnothing$;
let $\hat{G} := \textsc{Dual}_k(J, F, K)$;
for $n = 1, 2, \dots$ **do**
  let $D := D \cup \{d_n\}$; $K := \textsc{Sub}(D)$;
  if $D \nsubseteq \mathcal{L}(\hat{G})$ **then**
    let $J := \textsc{Con}(D)$ and $F := \textsc{Fun}(D)$;
  end if
  output $\hat{G} = \textsc{Dual}_k(J, F, K)$ as G_n;
end for

The conjecture grammar $\textsc{Dual}_k(J, F, K) = \langle N, \sigma, F, P, I\rangle$ of Algorithm 4 is defined from finite sets of contexts $J \subseteq \mathbb{C}$, functions $F \subseteq \mathbb{F}$ and substructures $K \subseteq \mathbb{S}$. For each $C \subseteq J_O$, we write $C^{(K)}$ to mean $C^\dagger \cap K_O$. This set can be seen as a finite approximation of $C^\dagger$, which is computable with MQs.

- $N_O = \{\, [\![C]\!] \mid C \subseteq J_O \text{ with } 1 \leq |C| \leq k \,\}$ for $O \in \Omega_{|D}$,
- $I = \{\, [\![\{\Box_*\}]\!] \,\}$,
- P consists of the rules of the form

$$[\![C_0]\!] \leftarrow f\langle [\![C_1]\!], \dots, [\![C_n]\!]\rangle$$

  where $f \in F_{O_0,\dots,O_n}$ for $[\![C_i]\!] \in N_{O_i}$ if $C_0 \odot f(C_1^{(K)}, \dots, C_m^{(K)}) \subseteq L_*$.

[1]We adopt the definition by Yoshinaka, which is slightly weaker than Clark's.

We say that a rule $[\![C_0]\!] \leftarrow f([\![C_1]\!], \dots, [\![C_n]\!])$ is *incorrect* if $C_0 \odot f(C_1^\dagger, \dots, C_n^\dagger) \nsubseteq L_*$. In that case, there are $s_i \in C_i^\dagger$ such that $C_0 \odot f(s_1, \dots, s_n) \nsubseteq L_*$.

Lemma 9. *For every finite $J \subseteq \mathbb{C}$ and $F \subseteq \mathbb{F}$ there is $K \subseteq \mathbb{S}$ such that $\hat{G} = \textsc{Dual}(J, F, K)$ has no incorrect rules and $|K| \leq p|F||J|^{k(p+1)}$, in which case $\mathcal{L}(\hat{G}) \subseteq L_*$.*

Let $G_* = \langle N_*, \sigma_*, F_*, P_*, I_*\rangle$ generate L_* and C_X a k-context of each nonterminal $X \in N_*$.

Lemma 10. *There is a finite subset $D \subseteq L_*$ such that $C_{|D} \supseteq C_X$ for all $X \in N_*$, $F_{|D} \supseteq F_*$ and $|D| \leq k|N_*| + |P_*|$. Moreover, if $J \supseteq C_X$ for all $X \in N_*$ and $F \supseteq F_*$, then $L_* \subseteq \mathcal{L}(\hat{G})$.*

6.5 Context-deterministic grammars

Definition 5 (Shirakawa and Yokomori (1993), Yoshinaka (2012)[2])**.** A Σ-grammar G is said to be *(weakly) context-deterministic* if every $c \in \mathcal{C}(G, X)$ is a 1-context of every $X \in N_O$.

Differently from Theorem 3, we do not need Assumption 4 for learning context-deterministic grammars with a minimally adequate teacher.

Theorem 5. *Under Assumptions 1, 2 and 3, Algorithm 3 learns any language L_* generated by a context-deterministic Σ-grammar G_* with a minimally adequate teacher in time polynomial in $|N_*|, |F_*|, \ell$ where ℓ is the total size of counterexamples given to the learner.*

Proof. By Lemmas 11, 12 and 13 below. □

Algorithm 5 uses the same grammar construction DUAL as Algorithm 4. By Lemma 9, when the oracle answers a negative counterexample d towards an EQ, our conjecture $\hat{G}$ must use an incorrect rule to derive d. To find and remove such an incorrect rule, Algorithm 5 calls a subroutine WITNESSD with a derivation tree τ_d of $\hat{G}$ whose yield is d. The procedure WITNESSD returns a finite set of substructures that witnesses an incorrect rule that contributes to generating d. An input given to WITNESSD is in general a $[\![c]\!]$-derivation tree τ such that $c \odot \tilde{\tau} \notin L_*$. Let $\tau = \rho[\tau_1, \dots, \tau_n]$ where $\rho = [\![c]\!] \leftarrow f\langle [\![c_1]\!], \dots, [\![c_n]\!]\rangle$. If there is i such that $c_i \odot \tilde{\tau}_i \notin L_*$, we recursively call WITNESSD(τ_i).

[2]We adopt the definition by Yoshinaka, which is slightly weaker than Shirakawa and Yokomori's.

Algorithm 5 Learning context-deterministic Σ-grammars

let $J := F := K := \varnothing$;
let $\hat{G} := \textsc{Dual}_1(J, F, K)$;
for $n = 1, 2, \ldots$ **do**
  if $\mathcal{L}(\hat{G}) = L_*$ (equivalence query) **then**
    output $\hat{G}$ and halt;
  else if the given counterexample d is positive ($d \in L_* - \mathcal{L}(\hat{G})$) **then**
    let $J := J \cup \textsc{Con}(d)$ and $F := F \cup \textsc{Fun}(d)$;
  else
    let $K := K \cup \textsc{WitnessD}(\tau)$ where τ is an (implicit) parse tree of d by $\hat{G}$
  end if
  let $\hat{G} = \textsc{Dual}_1(J, F, K)$;
end for

Otherwise, $\tilde{\tau}_i \in {c_i}^\dagger$ for all i, which means the rule ρ is incorrect. $\textsc{WitnessD}(\tau)$ returns the set $\{\, \tilde{\tau}_1, \ldots, \tilde{\tau}_n \,\}$. Differently from the case of WitnessP, an instance of a recursive call is always an (implicit) derivation tree of some $s \in \mathbb{S}_{|d}$. This explains why we do not need Assumption 4 in this case.

Lemma 11. *Time between an* EQ *and another is polynomially bounded.*

Lemma 12. *Each time Algorithm 5 receives a negative counterexample, at least one incorrect rule is removed.*

Lemma 13. *Let $G_* = \langle N_*, \sigma, F_*, P_*, I_* \rangle$ be a context-deterministic grammar for L_*. Each time Algorithm 5 receives a positive counterexample, the set $\{\, X \in N_* \mid J \cap \mathcal{C}(G_*, X) = \varnothing \,\} \cup (F_* - F)$ gets shrunk.*

6.6 Combined approaches

By combining primal and dual approaches, one can obtain stronger approaches (Yoshinaka, 2012). The class of Σ-grammars whose nonterminals admit either a k-kernel or l-context can be learned by combining the techniques presented in Sections 6.2 and 6.4 under Assumptions 1, 2 and 3. Also Σ-grammars whose nonterminals satisfy either the requirement to be congruential or to be context-deterministic can be learned with a minimally adequate teacher under Assumptions 1, 2, 3 and 4 (Sections 6.3 and 6.5).

7 Restricted cases

In some grammar classes, it may be the case that only (supersets of) $\mathbb{C}_{|d}$ and $\mathbb{F}_{|d}$ are computable in polynomial-time but $\mathbb{S}_{|d}$ is not, or the other way around: $\mathbb{S}_{|d}$ and $\mathbb{F}_{|d}$ are efficiently computable but $\mathbb{C}_{|d}$ is not. For example, in non-permuting parallel multiple CFGs (Seki et al., 1991), elements of $\mathbb{S}_{|d}$ for a string d are tuples of strings of the form $\langle v_1, \ldots, v_m \rangle$ for $d = u_0 v_1 u_1 \ldots v_m u_m$ and such substrings are polynomially many if m is fixed. However, $\mathbb{C}_{|d}$ contains exponentially many contexts. Clark and Yoshinaka (2014) showed that still a dual approach works for parallel multiple CFGs if nonterminals are known to have k-contexts belonging to a certain subset $\mathbf{C} \subseteq \mathbb{C}$ such that $\mathbf{C}_{|d} = \mathbb{C}_{|d} \cap \mathbf{C}$ is polynomial-time computable. A symmetric result of a primal approach has also been obtained by Kanazawa and Yoshinaka (2015) targeting a certain kind of tree grammars. This section does not postulate Assumption 3.

Definition 6. A Σ-grammar G is said to have the $(k, \mathbf{S})$-FKP if every nonterminal admits a k-kernel which is a subset of $\mathbf{S}$.

Assumption 5. *There are polynomial-time algorithms that compute* $\textsc{Sub}(d)$, $\textsc{Con}(d)$ *and* $\textsc{Fun}(d)$ *such that* $\mathbf{S}_{|d} \subseteq \textsc{Sub}(d) \subseteq \mathbb{S}$, $\mathbb{C}_{|d} \subseteq \textsc{Con}(d) \subseteq \mathbb{C}$ *and* $\mathbb{F}_{|d} \subseteq \textsc{Fun}(d) \subseteq \mathbb{F}$, *where* $\mathbf{S}_{|d} = \mathbf{S} \cap \mathbb{S}_{|d}$.

It is not hard to see that Algorithm 2 works for learning Σ-grammars with $(k, \mathbf{S})$-FKP under Assumptions 1, 2 and 5. All discussions in Section 6.2 hold for this restricted case.

The symmetric definition and assumption are as follows.

Definition 7. A Σ-grammar G is said to have the $(k, \mathbf{C})$-FCP if every nonterminal admits a k-context which is a subset of $\mathbf{C}$.

Assumption 6. *There are polynomial-time algorithms that compute* $\textsc{Sub}(d)$, $\textsc{Con}(d)$ *and* $\textsc{Fun}(d)$ *such that* $\mathbb{S}_{|d} \subseteq \textsc{Sub}(d) \subseteq \mathbb{S}$, $\mathbf{C}_{|d} \subseteq \textsc{Con}(d) \subseteq \mathbb{C}$ *and* $\mathbb{F}_{|d} \subseteq \textsc{Fun}(d) \subseteq \mathbb{F}$.

It is not hard to see that under Assumptions 1, 2 and 6, Algorithms 4 work for learning Σ-grammars with $(k, \mathbf{C})$-FCP Σ-grammars. All discussions in Section 6.4 hold for this restricted case.

When learning substitutable languages, even a weaker assumption suffices.

Assumption 7. *There are sets* $\mathbf{S} \subseteq \mathbb{S}$ *and* $\mathbf{C} \subseteq \mathbb{C}$ *such that for every nonterminal* X *of* $G \in \mathcal{G}(\Sigma)$*, we have* $\mathcal{S}(G, X) \cap \mathbf{S} \neq \varnothing$ *and* $\mathcal{C}(G, X) \cap \mathbf{C} \neq \varnothing$*. Moreover, there are polynomial-time algorithms that compute* $\textsc{Sub}(d)$*,* $\textsc{Con}(d)$ *and* $\textsc{Fun}(d)$ *such that* $\mathbf{S}_{|d} \subseteq \textsc{Sub}(d) \subseteq \mathbb{S}$*,* $\mathbf{C}_{|d} \subseteq \textsc{Con}(d) \subseteq \mathbb{C}$ *and* $\mathbb{F}_{|d} \subseteq \textsc{Fun}(d) \subseteq \mathbb{F}$*.*

Under Assumptions 1, 2 and 7, Algorithm 1 works using either SUBSTP or SUBSTD.

On the other hand, the results on the polynomial-time MAT learnability of congruential and context-deterministic Σ-grammars do not hold anymore under any of Assumptions 5, 6 and 7.

8 Extending learnable classes

This section compares learnable classes of Σ-languages for different Σ with the same special sort O_*. For Σ_1 and Σ_2 with $\Sigma_i = \langle \Omega^i, \mathbb{F}^i, O_* \rangle$, if $\Omega^1 \subseteq \Omega^2$ and $\mathbb{F}^1 \subseteq \mathbb{F}^2$, every Σ_1-grammar is a Σ_2-grammar, so $\mathcal{L}(\Sigma_1) \subseteq \mathcal{L}(\Sigma_2)$. However, since the distributional properties defined so far are relative to a signature, a Σ_1-grammar with a distributional property under Σ_1 does not necessarily have the corresponding property under Σ_2. Yet if $\mathbb{S}_O$ and $\mathbb{C}_O$ are preserved by moving from Σ_1 to Σ_2, the distributional properties other than the substitutability are preserved.

Let us define the direct union $\Sigma_0 = \langle \Omega^0, \mathbb{F}^0, O_* \rangle$ of arbitrary signatures Σ_1 and Σ_2 by $\Omega^0 = \{O_*\} \cup \{ (O, i) \mid O \in \Omega^i \text{ with } i \in \{1, 2\} \}$ where $\mathbb{O}_{(O,i)} = \{ (s, i) \mid s \in O \}$ and $\mathbb{F}^0 = \mathbb{G}^1 \cup \mathbb{G}^2 \cup \{\Box_1, \Box_2\}$, where $\mathbb{G}^i$ is a trivial variant of $\mathbb{F}^i$ working on the new domain and codomain of the form (O, i) and $\Box_i(s, i) = s$ for all $s \in O_*$. Then every Σ_i-grammar G can be seen as a special type of Σ_0-grammar by adding a new initial symbol Z and rules of the form $Z \leftarrow \Box_{\Sigma_i}\langle X \rangle$ for all initial symbols X of G. We have $\mathcal{L}(\Sigma_1) \cup \mathcal{L}(\Sigma_2) \subseteq \mathcal{L}(\Sigma_0)$. Every Σ_i-grammar that is congruential, context-deterministic, with the k-FKP or with the k-FCP for $i = 1, 2$ can be seen as a Σ_0-grammar with those properties. Note that $\mathbb{C}_{O_*}$ is the singleton of the identity function in Σ_0, which means any element of $\mathcal{L}(G)$ is a 1-kernel of the new initial symbol Z. In this way, from two signatures, one can obtain a richer learnable class of languages.

The above argument on signature generalization does not hold for substitutable case. Rather the opposite holds. If $\Omega^1 \subseteq \Omega^2$ and $\mathbb{F}^1 \subseteq \mathbb{F}^2$, then a language substitutable under Σ_2 is substitutable under Σ_1 but not vice versa.

Let us say that Σ_2 is *finer* than Σ_1 if every sort of Ω^1 is partitioned into finite number of sorts in Ω^2 and every function of $\mathbb{F}^2$ is a subfunction of some function in $\mathbb{F}^1$ which accords with the partition. That is, every sort O of Ω^1 has a finite set $\Omega^2_O \subseteq \Omega^2$ such that $O = \bigcup \Omega^2_O$ and $\mathbb{F}^1_{O_0,\ldots,O_n} = \bigcup \{ \mathbb{F}^2_{O'_0,\ldots,O'_n} \mid O'_i \in \Omega^2_{O_i} \}$. For instance, $\Sigma_{k,l}$ is finer than $\Sigma_{k',l'}$ for $k' \leq k$ and $l' \leq l$ in Example 4. If Σ_2 is finer than Σ_1, $\mathcal{L}(\Sigma_1) = \mathcal{L}(\Sigma_2)$ holds. Every language substitutable under Σ_1 is substitutable under Σ_2 but not vice versa. Moreover, every congruential (resp. context-deterministic) Σ_1-grammar has an equivalent congruential (resp. context-deterministic) Σ_2-grammar but not vice versa.

9 Grammars with partial functions

Yoshinaka (2015) showed that a dual approach can be applied to the learning of *conjunctive grammars*. Conjunctive grammars (Okhotin, 2001) are CFGs extended with the conjunctive operation $\&$ so that one can extract the intersection of the languages of nonterminals. For example, a conjunctive rule $A_0 \to A_1 \& A_2$ means that if *both* A_1 and A_2 generate the same string u then so does A_0. Conjunctive grammars cannot be seen as Σ-grammars, since the conjunctive operation $\&$ is a *partial* function whose domain is not represented as the direct product of two sorts, which is not legitimate in the general framework of Σ-grammars.

A *partial signature* is a triple $\Pi = \langle \Omega, \mathbb{F}, O_* \rangle$ which is defined in the way similar to a (total) signature but $\mathbb{F}$ may have partial functions. Accordingly contexts in $\mathbb{C}$ will be partial functions. We do not have $\mathcal{C}(G, X) \odot \mathcal{S}(G, X) \subseteq \mathcal{L}(G)$ any more, since $c \odot s$ may not be defined for some elements $c \in \mathcal{C}(G, X)$ and $s \in \mathcal{S}(G, X)$. The correspondence between O-concept lattices and Σ-grammars collapses. This prevents the application of the theory of distributional learning developed in this paper to Π-grammars. Still we can generalize the discussion on the learning of conjunctive grammars.

Definition 8. A Π-grammar G is said to have the *strong* k-FCP if for any $X \in N_O$, there is a finite set

$C_X \subseteq \mathbb{C}_O$ with $|C_X| \leq k$ such that

$$\mathcal{S}(G, X) = \{\, s \mid c \odot s \in L \text{ for all } c \in C_X \,\}.$$

Definition 8 requires every $c \in C_X$ to be total on $\mathcal{S}(G, X)$. One can learn Π-grammars with the strong k-FCP under Assumptions 1, 2 and 6, where $\mathbf{C}$ consists of total functions only. The grammar construction DUAL_k should be modified so that we have a rule $[\![C_0]\!] \leftarrow f\langle [\![C_1]\!], \ldots, [\![C_n]\!] \rangle$ if $c \odot f(s_1, \ldots, s_n) \in L_*$ for any $c \in C_0$ and $s_i \in C_i^{(K)}$ such that $f(s_1, \ldots, s_n)$ is defined. One might think that one can naturally define context-deterministic grammars accordingly: Every $c \in \mathcal{C}(G, X)$ should be a 1-context of X. However, this means that functions in such a Π-grammar are essentially total.

Acknowledgments

The view presented in this paper has been sharpened through the interactions and discussions with several researchers with whom I worked on the distributional learning of generalized CFGs. I would like to show my deepest gratitude to Alexander Clark, Makoto Kanazawa, Anna Kasprzik and Gregory Kobele. Without those people this work would have been hard to accomplish. Any insufficiency or errors in this paper are of course of my own.

References

Alexander Clark and Rémi Eyraud. 2007. Polynomial identification in the limit of substitutable context-free languages. *Journal of Machine Learning Research*, 8:1725–1745.

Alexander Clark and Ryo Yoshinaka. 2014. Distributional learning of parallel multiple context-free grammars. *Machine Learning*, 96(1-2):5–31.

Alexander Clark, Rémi Eyraud, and Amaury Habrard. 2009. A note on contextual binary feature grammars. In *EACL 2009 workshop on Computational Linguistic Aspects of Grammatical Inference*, pp. 33–40.

Alexander Clark. 2010a. Distributional learning of some context-free languages with a minimally adequate teacher. In J. Sempere and P. García, editors, *ICGI*, LNCS 6339, pp. 24–37. Springer.

Alexander Clark. 2010b. Learning context free grammars with the syntactic concept lattice. In J. Sempere and P. García, editors, *ICGI*, LNCS 6339, pp. 38–51. Springer.

Alexander Clark. 2010c. Towards general algorithms for grammatical inference. In M. Hutter, F. Stephan, V. Vovk, and T. Zeugmann, editors, *ALT*, LNCS 6331, pp. 11–30. Springer.

Yuichi Kaji, Ryuichi Nakanishi, Hiroyuki Seki, and Tadao Kasami. 1992. The universal recognition problems for parallel multiple context-free grammars and for their subclasses. *IEICE Transaction on Information and Systems*, E75-D(7):499–508.

Makoto Kanazawa and Ryo Yoshinaka. 2015. Distributional learning and context/substructure enumerability in non-linear tree grammars. In *Formal Grammar - 20th International Conference, FG 2015, Barcelona, Spain, August 8-9, 2015. Proceedings*. to appear.

Anna Kasprzik and Ryo Yoshinaka. 2011. Distributional learning of simple context-free tree grammars. In J. Kivinen, C. Szepesvári, E. Ukkonen, and T. Zeugmann, editors, *ALT*, LNCS 6925, pp. 398–412. Springer.

Alexander Okhotin. 2001. Conjunctive grammars. *Journal of Automata, Languages and Combinatorics*, 6(4):519–535.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191–229.

Hiromi Shirakawa and Takashi Yokomori. 1993. Polynomial-time MAT learning of c-deterministic context-free grammars. *Transaction of Information Processing Society of Japan*, 34:380–390.

Ryo Yoshinaka and Makoto Kanazawa. 2011. Distributional learning of abstract categorial grammars. In S. Pogodalla and J.-P. Prost, editors, *LACL*, LNCS 6736, pp. 251–266. Springer.

Ryo Yoshinaka. 2008. Identification in the limit of k, l-substitutable context-free languages. In A. Clark, F. Coste, and L. Miclet, editors, *ICGI*, LNCS 5278, pp. 266–279. Springer.

Ryo Yoshinaka. 2011a. Efficient learning of multiple context-free languages with multidimensional substitutability from positive data. *Theoretical Computer Science*, 412(19):1821–1831.

Ryo Yoshinaka. 2011b. Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In G. Mauri and A. Leporati, editors, *DLT*, LNCS 6795, pp. 429–440. Springer.

Ryo Yoshinaka. 2012. Integration of the dual approaches in the distributional learning of context-free grammars. In A. H. Dediu and C. Martín-Vide, editors, *LATA*, LNCS 7183, pp. 538–550. Springer.

Ryo Yoshinaka. 2015. Learning conjunctive grammars and contextual binary feature grammars. In A. H. Dediu, E. Formenti, C. Martín-Vide, and B. Truthe, editors, *LATA*, LNCS 8977, pp. 623–635. Springer.