# Polynomial Time Learning of Some Multiple Context-Free Languages with a Minimally Adequate Teacher

Ryo Yoshinaka[1]* and Alexander Clark[2]

[1] MINATO Discrete Structure Manipulation System Project,
ERATO, Japan Science and Technology Agency
ryoshinaka@erato.ist.hokudai.ac.jp
[2] Department of Computer Science, Royal Holloway, University of London
alexc@cs.rhul.ac.uk

**Abstract.** We present an algorithm for the inference of some Multiple Context-Free Grammars from Membership and Equivalence Queries, using the Minimally Adequate Teacher model of Angluin. This is an extension of the congruence based methods for learning some Context-Free Grammars proposed by Clark (ICGI 2010). We define the natural extension of the syntactic congruence to tuples of strings, and demonstrate we can efficiently learn the class of Multiple Context-Free Grammars where the non-terminals correspond to the congruence classes under this relation.

## 1 Introduction

In this paper we look at efficient algorithms for the inference of multiple context free grammars (MCFGs). MCFGs are a natural extension of context free grammars (CFGs), where the non-terminal symbols derive tuples of strings, which are then combined with each other in a limited range of ways. Since some natural language phenomena were found not to be context-free, the notion of *mildly context-sensitive languages* was proposed and studied for better describing natural languages, while keeping tractability [1]. MCFGs are regarded as a representative mildly context-sensitive formalism, which has many equivalent formalisms such as linear context-free rewriting systems [2], multicomponent tree adjoining grammars [3, 4], minimalist grammars [5], hyperedge replacement grammars [6], etc.

In recent years, there has been rapid progress in the field of context-free grammatical inference using techniques of distributional learning. The first result along these lines was given by Clark and Eyraud [7]. They showed a polynomial

* He is concurrently working in Graduate School of Information Science and Technology, Hokkaido University. This work was supported in part by Grant-in-Aid for Young Scientists (B-20700124) from the Ministry of Education, Culture, Sports, Science and Technology of Japan.

identification in the limit result from positive data alone for a class of languages called the substitutable context free languages.

We assume we have a finite non-empty alphabet $\Sigma$ and an extra hole or gap symbol, $\Box$, which is not in $\Sigma$. A context is a string with one hole in, written as $l\Box r$, and the distribution of a string $u$ in a language $L$ is defined as $L/u = \{\, l\Box r \mid lur \in L \,\}$. There is a natural congruence relation called the syntactic congruence which is defined as $u \equiv_L v$ iff $L/u = L/v$.

The learning approach of Clark and Eyraud is based on the following observations:

**Lemma 1.** *If $u \equiv_L u'$ then $uw \equiv_L u'w$. If $u \equiv_L u'$ and $v \equiv_L v'$ then $uv \equiv_L u'v'$.*

This means that it is trivial to construct a CFG where the non-terminals correspond to congruence classes under this relation, as we can construct rules of the form $[uv] \to [u], [v]$. All rules of this form will be valid since $[u][v] \subseteq [uv]$ by Lemma 1.

Let us consider now the language $L_c = \{\, wcwc \mid w \in \{a, b\}^* \,\}$. This is a slight variant of the classic copy language which is not context-free. It is formally similar to the constructions in Swiss German which established that the class of natural languages is not included in the class of context-free languages and is thus a classic test case for linguistic representation [8,9].

We will extend the approach by considering relationships not between strings, but between tuples of strings. Here for concreteness we will consider the case of tuples of dimension 2, i.e. pairs; but in the remainder of the paper we will define this for tuples of arbitrary dimension.

Let $\langle u, v\rangle$ be an ordered pair of strings. We define $L/\langle u, v\rangle = \{\, l\Box m\Box r \mid lumvr \in L \,\}$ and we then define an equivalence relation between pairs of strings which is analogous to the syntactic congruence: $\langle u, v\rangle \equiv_L \langle u', v'\rangle$ iff $L/\langle u, v\rangle = L/\langle u', v'\rangle$.

Note that when we consider $L_c$ it is easy to see that $\langle c, c\rangle \equiv_{L_c} \langle ac, ac\rangle \equiv_{L_c} \langle bc, bc\rangle$, but that $\langle a, a\rangle$ is not congruent to $\langle b, b\rangle$, since for example $\Box\Box caac$ is in $L/\langle a, a\rangle$ but not $L/\langle b, b\rangle$.

Just as with Lemma 1 above, this congruence has interesting properties.

**Lemma 2.** *If $\langle u, v\rangle \equiv_L \langle u', v'\rangle$ and $\langle x, y\rangle \equiv_L \langle x', y'\rangle$, then $\langle ux, vy\rangle \equiv_L \langle u'x', v'y'\rangle$.*

Note that there is now more than one way of combining these two elements. If they are single strings, then there are only two ways – $uv$ and $vu$. When there are multiple strings we can combine them in many different ways. For example, it is also the case that $\langle uxy, v\rangle \equiv_L \langle u'x'y', v'\rangle$, but in general, it is not the case that $\langle uyx, v\rangle \equiv_L \langle u'y'x', v'\rangle$.

In the specific case of Lemma 2 we can consider the concatenation operation as a function $f : (\Sigma^* \times \Sigma^*)^2 \to \Sigma^* \times \Sigma^*$ defined as $f(\langle u, v\rangle, \langle x, y\rangle) = \langle ux, vy\rangle$. If we use the notation $\boldsymbol{w}$ for tuples then the lemma says that if $\boldsymbol{u} \equiv_L \boldsymbol{u'}$ and $\boldsymbol{v} \equiv_L \boldsymbol{v'}$, then $f(\boldsymbol{u}, \boldsymbol{v}) \equiv_L f(\boldsymbol{u'}, \boldsymbol{v'})$.

We can then generalise Lemma 2 to all functions $f$ that satisfy a certain set of conditions; as we discuss formally later on, these must be linear regular and non-permuting.

Our learning algorithm will work by constructing an MCFG where each non-terminal generates a congruence class of tuples of strings.

For our learning model, we will use in this paper the Minimally Adequate Teacher (MAT) model introduced by Angluin in [10]. This model assumes the existence of a teacher who can answer two sorts of queries: first, the learner can ask Membership Queries (MQs), where the learner queries whether a given string is in the target language or not, and secondly the learner can ask Equivalence Queries (EQs), where the learner queries whether a particular hypothesis is correct or not; if the hypothesis is not correct, then the teacher will give a counter-example to the learner. The learner is required to use only a limited amount of computation before terminating – in particular there must be a polynomial $p$ such that the total amount of computation used is less than $p(n, \ell)$, where $n$ is the size of the target representation, and $\ell$ is the length of the longest counter-example returned by the teacher.

This model is rather abstract; but it has a number of important advantages that make it appropriate for this paper. First of all, it is a very restrictive model. There are strong negative results [11] that show that certain natural classes – regular grammars, CFGs are not learnable under this MAT paradigm. Secondly, it is also permissive – it allows reasonably large classes of languages to be learned. Angluin's famous LSTAR algorithm for the inference of regular languages is a classic and well-studied algorithm, and one of the great success stories of grammatical inference.

Finally, if we look at the history of regular and context-free grammatical inference it appears to work well as a substitute for more realistic probabilistic models. Probabilistic DFAs turn out to be PAC-learnable under a very rigorous model [12], albeit stratified by certain parameters. Thus though this model is unrealistic it seems a good intermediate step, and it is technically easy to work with.

The main result of this paper is a polynomial MAT learning algorithm for a class of MCFGs. In Section 2 we will introduce our notation. Our learning algorithm is discussed in Section 3. We conclude this paper in Section 4.

## 2 Preliminaries

### 2.1 Basic Definitions and Notations

The set of non-negative integers is denoted by $\mathbb{N}$ and this paper will consider only numbers in $\mathbb{N}$. The cardinality of a set $S$ is denoted by $|S|$. If $w$ is a string over an alphabet $\Sigma$, $|w|$ denotes its length. $\varnothing$ is the empty set and $\lambda$ is the empty string. $\Sigma^*$ denotes the set of all strings over $\Sigma$. We write $\Sigma^+ = \Sigma^* - \{\lambda\}$, $\Sigma^k = \{ w \in \Sigma^* \mid |w| = k \}$ and $\Sigma^{\leq k} = \{ w \in \Sigma^* \mid |w| \leq k \}$. Any subset of $\Sigma^*$ is called a *language (over $\Sigma$)*. If $L$ is a finite language, its size is defined as $\|L\| = |L| + \sum_{w \in L} |w|$. An *$m$-word* is an $m$-tuple of strings and we denote the set of $m$-words by $(\Sigma^*)^{\langle m \rangle}$. Similarly we define $(\cdot)^{\langle * \rangle}$, $(\cdot)^{\langle + \rangle}$, $(\cdot)^{\langle \leq m \rangle}$. Any $m$-word is called a *multiword*. Thus $(\Sigma^*)^{\langle * \rangle}$ denotes the set of all multiwords. For $\boldsymbol{w} = \langle w_1, \dots, w_m \rangle \in (\Sigma^*)^{\langle m \rangle}$, $|\boldsymbol{w}|$ denotes its length $m$ and $\|\boldsymbol{w}\|$ denotes its *size* $m + \sum_{1 \leq i \leq m} |w_i|$. We use the symbol $\square$, assuming that $\square \notin \Sigma$, for representing a "hole", which is supposed to be replaced by another string. We write $\Sigma_\square$ for $\Sigma \cup \{\square\}$. A string $\mathsf{x}$ over $\Sigma_\square$ is called an *$m$-context* if $\mathsf{x}$ contains $m$ occurrences of $\square$. $m$-contexts are also called *multicontexts*. For an $m$-context $\mathsf{x} = x_0 \square x_1 \square \dots \square x_m$ with $x_0, \dots, x_m \in \Sigma^*$ and an $m$-word $\boldsymbol{y} = \langle y_1, \dots, y_m \rangle \in (\Sigma_\square^*)^{\langle m \rangle}$, we define

$$\mathsf{x} \odot \boldsymbol{y} = x_0 y_1 x_1 \dots y_n x_n$$

and say that $\boldsymbol{y}$ is a *sub-multiword* of $\mathsf{x} \odot \boldsymbol{y}$. Note that $\square$ is the empty context and we have $\square \odot \langle y \rangle = y$ for any $y \in \Sigma^*$. For $L \subseteq \Sigma^*$ and $p \geq 1$, we define

$$\mathcal{S}^{\leq p}(L) = \{\, \boldsymbol{y} \in (\Sigma^+)^{\langle \leq p \rangle} \mid \mathsf{x} \odot \boldsymbol{y} \in L \text{ for some } \mathsf{x} \in \Sigma_\square^* \,\},$$
$$\mathcal{C}^{\leq p}(L) = \{\, \mathsf{x} \in \Sigma_\square^* \mid \mathsf{x} \odot \boldsymbol{y} \in L \text{ for some } \boldsymbol{y} \in (\Sigma^+)^{\langle \leq p \rangle} \,\},$$

and for $\boldsymbol{y} \in (\Sigma^*)^{\langle * \rangle}$, we define

$$L/\boldsymbol{y} = \{\, \mathsf{x} \in \Sigma_\square^* \mid \mathsf{x} \odot \boldsymbol{y} \in L \,\}.$$

Obviously computation of $\mathcal{S}^{\leq p}(L)$ can be done in $O(\|L\|^{2p})$ time if $L$ is finite.

### 2.2 Linear Regular Functions

Let us suppose a countably infinite set $Z$ of *variables* disjoint from $\Sigma$. A function $f$ from $(\Sigma^*)^{\langle m_1 \rangle} \times \dots \times (\Sigma^*)^{\langle m_n \rangle}$ to $(\Sigma^*)^{\langle m \rangle}$ is said to be *linear*, if there is $\langle \alpha_1, \dots, \alpha_m \rangle \in ((\Sigma \cup \{\, z_{ij} \in Z \mid 1 \leq i \leq n,\ 1 \leq j \leq m_i \,\})^*)^{\langle m \rangle}$ such that each variable $z_{ij}$ occurs at most once in $\langle \alpha_1, \dots, \alpha_m \rangle$ and

$$f(\boldsymbol{w}_1, \dots, \boldsymbol{w}_n) = \langle \alpha_1[\boldsymbol{z} := \boldsymbol{w}], \dots, \alpha_m[\boldsymbol{z} := \boldsymbol{w}] \rangle$$

for any $\boldsymbol{w}_i = \langle w_{i1}, \dots, w_{im_i} \rangle \in (\Sigma^*)^{\langle m_i \rangle}$ with $1 \leq i \leq n$, where $\alpha_k[\boldsymbol{z} := \boldsymbol{w}]$ denotes the string obtained by replacing each variable $z_{ij}$ with the string $w_{ij}$. We call $f$ *linear regular* if every variable $z_{ij}$ occurs in $\langle \alpha_1, \dots, \alpha_m \rangle$. We say that $f$ is *$\lambda$-free* when no $\alpha_k$ from $\langle \alpha_1, \dots, \alpha_m \rangle$ is $\lambda$. A linear regular function $f$ is said to be *non-permuting*, if $z_{ij}$ always occurs to the left of $z_{i(j+1)}$ in $\alpha_1 \dots \alpha_m$ for $1 \leq i \leq n$ and $1 \leq j < m_i$. The *rank* $\mathrm{rank}(f)$ of $f$ is defined to be $n$ and the *size* $\mathrm{size}(f)$ of $f$ is $\|\langle \alpha_1, \dots, \alpha_m \rangle\|$.

*Example 1.* Among the functions defined below, where $a, b \in \Sigma$, $f$ is not linear, while $g$ and $h$ are linear regular. Moreover $h$ is $\lambda$-free and non-permuting.

$$f(\langle z_{11}, z_{12} \rangle, \langle z_{21} \rangle) = \langle z_{11} z_{12}, z_{11} z_{21} z_{11} \rangle,$$
$$g(\langle z_{11}, z_{12} \rangle, \langle z_{21} \rangle) = \langle z_{12}, z_{11} b z_{21}, \lambda \rangle,$$
$$h(\langle z_{11}, z_{12} \rangle, \langle z_{21} \rangle) = \langle a, z_{11} b z_{21}, z_{12} \rangle.$$

The following lemma is the generalization of Lemmas 1 and 2 mentioned in the introduction, on which our approach is based.

**Lemma 3.** *For any language $L \subseteq \Sigma^*$ and any $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in (\Sigma^*)^{\langle * \rangle}$ such that $|\boldsymbol{u}_i| = |\boldsymbol{v}_i|$ and $L/\boldsymbol{u}_i = L/\boldsymbol{v}_i$ for all $i$, we have $L/f(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n) = L/f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$ for any non-permuting linear regular function $f$.*

*Proof.* Let $m_i = |\boldsymbol{u}_i| = |\boldsymbol{v}_i|$. Suppose that $\mathsf{x} \in L/f(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n)$, i.e., $\mathsf{x} \odot f(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n) \in L$. The following inference is allowed:

$$\mathsf{x} \odot f(\square^{\langle m_1 \rangle}, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n) \in L/\boldsymbol{u}_1 = L/\boldsymbol{v}_1 \implies \mathsf{x} \odot f(\boldsymbol{v}_1, \boldsymbol{u}_2, \dots, \boldsymbol{u}_n) \in L$$
$$\implies \mathsf{x} \odot f(\boldsymbol{v}_1, \square^{\langle m_2 \rangle}, \boldsymbol{u}_3, \dots, \boldsymbol{u}_n) \in L/\boldsymbol{u}_2 = L/\boldsymbol{v}_2 \implies$$
$$\mathsf{x} \odot f(\boldsymbol{v}_1, \boldsymbol{v}_2, \boldsymbol{u}_3, \dots, \boldsymbol{u}_n) \in L \implies \dots \implies \mathsf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n) \in L.$$

Hence $\mathsf{x} \in L/f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$. $\square$

## 2.3 Multiple Context-Free Grammars

A *multiple context-free grammar (*MCFG*)* is a tuple $G = \langle \Sigma, V_{\dim}, F, P, I \rangle$, where

- $\Sigma$ is a finite set of *terminal symbols*,
- $V_{\dim} = \langle V, \dim \rangle$ is the pair of a finite set $V$ of *nonterminal symbols* and a function dim giving a positive integer, called a *dimension*, to each element of $V$,
- $F$ is a finite set of *linear functions*,[3]
- $P$ is a finite set of *rules* of the form $A \to f(B_1, \dots, B_n)$ where $A, B_1, \dots, B_n \in V$ and $f \in F$ maps $(\Sigma^*)^{\langle \dim(B_1) \rangle} \times \dots \times (\Sigma^*)^{\langle \dim(B_n) \rangle}$ to $(\Sigma^*)^{\langle \dim(A) \rangle}$,
- $I$ is a subset of $V$ and all elements of $I$ have dimension 1. Elements of $I$ are called *initial symbols*.

We note that our definition of MCFGs is slightly different from the original [13], where grammars have exactly one initial symbol, but this change does not affect the generative capacity of MCFGs.

We will simply write $V$ for $V_{\dim}$ if no confusion occurs. If a rule has a function $f$, then its right hand side must have rank$(f)$ occurrences of nonterminals by definition. If rank$(f) = 0$ and $f() = \boldsymbol{v}$, we may write $A \to \boldsymbol{v}$ instead of $A \to f()$. If rank$(f) = 1$ and $f$ is the identity, we may write $A \to B$ instead of $A \to f(B)$, where $\dim(A) = \dim(B)$. The *size* $\|G\|$ of $G$ is defined as $\|G\| = |P| + \sum_{\rho \in P} \mathrm{size}(\rho)$ where $\mathrm{size}(A \to f(B_1, \dots, B_n)) = \mathrm{size}(f) + n + 1$.

For $A \in V$, *$A$-derivation trees* are recursively defined as follows:

- If a rule $\pi \in P$ has the form $A \to \boldsymbol{w}$, then $\pi$ is an $A$-derivation tree for $\boldsymbol{w}$. We call $\boldsymbol{w}$ the *yield* of $\pi$.
- If a rule $\pi \in P$ has the form $A \to f(B_1, \dots, B_n)$ and $t_i$ is a $B_i$-derivation tree for $\boldsymbol{w}_i$ for all $i = 1, \dots, n$, then the tree whose root is $\pi$ with immediate subtrees $t_1, \dots, t_n$ from left to right, which we write as $\pi(t_1, \dots, t_n)$, is an $A$-derivation tree for $f(\boldsymbol{w}_1, \dots, \boldsymbol{w}_n)$, which is called the *yield* of $\pi(t_1, \dots, t_n)$.

[3] We identify a function with its name for convenience.

– nothing else is an $A$-derivation tree.

For all $A \in V$ we define

$$\mathcal{L}(G, A) = \{\, \boldsymbol{w} \in (\Sigma^*)^{\langle \dim(A) \rangle} \mid \text{there is an } A\text{-derivation tree for } \boldsymbol{w} \,\}.$$

The *language* $\mathcal{L}(G)$ *generated by* $G$ means the set $\{\, w \in \Sigma^* \mid \langle w \rangle \in \mathcal{L}(G, S)$ with $S \in I \,\}$, which is called a *multiple context-free language (*MCFL*)*. If $S \in I$, an $S$-derivation tree for $\langle w \rangle$ is simply called a derivation tree for $w \in \mathcal{L}(G)$. Two grammars $G$ and $G'$ are *equivalent* if $\mathcal{L}(G) = \mathcal{L}(G')$.

This paper assumes that all linear functions in $F$ are linear regular, $\lambda$-free and non-permuting. In fact those assumptions do not affect their generative capacity modulo $\lambda$ [13, 14].

We denote by $\mathbb{G}(p, r)$ the collection of MCFGs $G$ whose nonterminals are assigned a dimension at most $p$ and whose functions have a rank at most $r$. Then we define $\mathbb{L}(p, r) = \{\, \mathcal{L}(G) \mid G \in \mathbb{G}(p, r) \,\}$, We also write $\mathbb{G}(p, *) = \bigcup_{r \in \mathbb{N}} \mathbb{G}(p, r)$ and $\mathbb{L}(p, *) = \bigcup_{r \in \mathbb{N}} \mathbb{L}(p, r)$. The class of context-free grammars (CFGs) is identified with $\mathbb{G}(1, *)$ and all CFGs in Chomsky normal form are in $\mathbb{G}(1, 2)$. Thus $\mathbb{L}(1, 2) = \mathbb{L}(1, *)$.

*Example 2.* Let $G$ be the MCFG $\langle \Sigma, V, F, P, \{S\} \rangle$ over $\Sigma = \{a, b, c, d\}$ whose rules are

$$\pi_1 : S \to f(A, B) \text{ with } f(\langle z_{11}, z_{12} \rangle, \langle z_{21}, z_{22} \rangle) = \langle z_{11} z_{21} z_{12} z_{22} \rangle,$$
$$\pi_2 : A \to g(A) \text{ with } g(\langle z_1, z_2 \rangle) = \langle a z_1, c z_2 \rangle, \qquad \pi_3 : A \to \langle a, c \rangle,$$
$$\pi_4 : B \to h(B) \text{ with } h(\langle z_1, z_2 \rangle) = \langle z_1 b, z_2 d \rangle, \qquad \pi_5 : B \to \langle b, d \rangle,$$

where $V = \{S, A, B\}$ with $\dim(S) = 1$, $\dim(A) = \dim(B) = 2$, and $F$ consists of $f$, $g$, $h$ and the constant functions appearing in the rules $\pi_3$ and $\pi_5$. One can see, for example, $aabccd \in \mathcal{L}(G)$ thanks to the derivation tree $\pi_1(\pi_2(\pi_3), \pi_5)$: $\langle a, c \rangle \in \mathcal{L}(G, A)$ by $\pi_3$, $\langle aa, cc \rangle \in \mathcal{L}(G, A)$ by $\pi_2$, $\langle b, d \rangle \in \mathcal{L}(G, B)$ by $\pi_5$ and $\langle aabccd \rangle \in \mathcal{L}(G, S)$ by $\pi_1$. We have $\mathcal{L}(G) = \{\, a^m b^n c^m d^n \mid m, n \geq 1 \,\}$.

Seki et al. [13] and Rambow and Satta [15] have investigated the hierarchy of MCFLs.

**Proposition 1 (Seki et al. [13], Rambow and Satta [15]).**
*For $p \geq 1$, $\mathbb{L}(p, *) \subsetneq \mathbb{L}(p + 1, *)$.*
*For $p \geq 2$, $r \geq 1$, $\mathbb{L}(p, r) \subsetneq \mathbb{L}(p, r + 1)$ except for $\mathbb{L}(2, 2) = \mathbb{L}(2, 3)$.*
*For $p \geq 1$, $r \geq 3$ and $1 \leq k \leq r - 2$, $\mathbb{L}(p, r) \subseteq \mathbb{L}((k + 1)p, r - k)$.*

**Theorem 1 (Seki et al. [13], Kaji et al. [16]).** *Let $p$ and $r$ be fixed. It is decidable in $O(\|G\|^2 |w|^{p(r+1)})$ time whether $w \in \mathcal{L}(G)$ for any* MCFG *$G \in \mathbb{G}(p, r)$ and $w \in \Sigma^*$.*

## 2.4 Congruential Multiple Context-Free Grammars

Now we introduce the languages of our learning target.

**Definition 1.** We say that an MCFG $G \in \mathbb{G}(p, r)$ is *p-congruential* if for every nonterminal $A$ and any $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{L}(G, A)$, it holds that $\mathcal{L}(G)/\boldsymbol{u} = \mathcal{L}(G)/\boldsymbol{v}$.

In a $p$-congruential MCFG ($p$-CMCFG or CMCFG for short), one can merge two nonterminals $A$ and $B$ without changing the language when $\mathcal{L}(G)/\boldsymbol{u} = \mathcal{L}(G)/\boldsymbol{v}$ for $\boldsymbol{u} \in \mathcal{L}(G, A)$ and $\boldsymbol{v} \in \mathcal{L}(G, B)$.

We let $\mathbb{CG}(p, r)$ denote the class of $p$-CMCFGs from $\mathbb{G}(p, r)$ and $\mathbb{CL}(p, r)$ the corresponding class of languages. Our learning target is $\mathbb{CL}(p, r)$ for each $p, r \geq 1$. The class $\mathbb{CL}(1, 2)$ corresponds to congruential CFLs introduced by Clark [17], which include all regular languages.

The grammar $G$ in Example 2 is 2-congruential. It is easy to see that for any $\langle a^m, c^m \rangle \in \mathcal{L}(G, A)$, we have

$$\mathcal{L}(G)/\langle a^m, c^m \rangle = \{\, a^i \Box a^j b^n c^k \Box c^l d^n \mid i + j = k + l \geq 0 \text{ and } n \geq 1 \,\},$$

which is independent of $m$. Similarly one sees that all elements of $\mathcal{L}(G, B)$ share the same set of multicontexts. Obviously $\mathcal{L}(G)/\langle a^m b^n c^m d^n \rangle = \{\Box\}$ for any $\langle a^m b^n c^m d^n \rangle \in \mathcal{L}(G, S)$.

Another example of a CMCFL is $L_c = \{\, wcwc \mid w \in \{a, b\}^* \,\} \in \mathbb{CL}(2, 1)$. On the other hand, neither $L_1 = \{\, a^m b^n \mid 1 \leq m \leq n \,\}$ nor $L_2 = \{\, a^n b^n \mid n \geq 1 \,\} \cup \{\, a^n b^{2n} \mid n \geq 1 \,\}$ is $p$-congruential for any $p$. We now explain why $L_2 \notin \mathbb{CL}(*, *)$. A similar discussion is applicable to $L_1$. If an MCFG generates $L_2$, it must have a nonterminal that derives multiwords $\boldsymbol{u}$ and $\boldsymbol{v}$ that have different number of occurrences of $a$. Then it is not hard to see that $\boldsymbol{u}$ and $\boldsymbol{v}$ do not share the same set of multicontexts. However $\{\, a^n b^n c \mid n \geq 1 \,\} \cup \{\, a^n b^{2n} d \mid n \geq 1 \,\} \in \mathbb{L}(1, 1) \cap \mathbb{CL}(2, 1) - \mathbb{CL}(1, *)$.

## 3 Learning of Congruential Multiple Context-Free Grammars with a Minimally Adequate Teacher

### 3.1 Minimally Adequate Teacher

Our learning model is based on Angluin's MAT learning [18]. A learner has a teacher who answers two kinds of queries from the learner: *membership queries (*MQ*s)* and *equivalence queries (*EQ*s)*. An instance of an MQ is a string $w$ over $\Sigma$ and the teacher answers "YES" if $w \in L_*$ and otherwise "NO", where we use $L_*$ to refer to the learning target. An instance of an EQ is a grammar $\hat{G}$ and the teacher answers "YES" if $\mathcal{L}(\hat{G}) = L_*$ and otherwise returns a counter-example $w \in (L_* - \mathcal{L}(\hat{G})) \cup (\mathcal{L}(\hat{G}) - L_*)$. If $w \in L_* - \mathcal{L}(\hat{G})$, then it is called a *positive counter-example*, and if $w \in \mathcal{L}(\hat{G}) - L_*$, it is called a *negative counter-example*. The teacher is supposed to answer every query in constant time. The learning process finishes when the teacher answers "YES" to an EQ. In this learning scheme, we fix a class $\mathbb{G}$ of grammars representing our learning targets and require a learner to output a correct grammar in polynomial time in $\|G_*\|$ and $\ell$ where $G_*$ is a smallest grammar in $\mathbb{G}$ such that $\mathcal{L}(G_*) = L_*$ and $\ell$ is the length of the longest counter-example given by the teacher.

We remark that we do not restrict instances of EQs to grammars in the class $\mathbb{G}$. Queries of this type are often called extended EQs to emphasize the difference from the restricted type of EQs.

### 3.2 Hypotheses

Hereafter we arbitrarily fix two natural numbers $p \geq 1$ and $r \geq 1$. Let $L_* \subseteq \Sigma^*$ be the target language from $\mathbb{CL}(p, r)$. Our learning algorithm computes grammars in $\mathbb{G}(p, r)$ from three parameters $K \subseteq \mathcal{S}^{\leq p}(L_*)$, $X \subseteq \mathcal{C}^{\leq p}(L_*)$ and $L_*$ where $K$ and $X$ are always finite. Of course we cannot take $L_*$ as a part of the input, but in fact a finite number of MQs is enough to construct the following MCFG $\mathcal{G}^r(K, X, L_*) = \langle \Sigma, V, F, P, I\rangle$. The set of nonterminal symbols is $V = K$ and we will write $[\![\boldsymbol{v}]\!]$ instead of $\boldsymbol{v}$ for clarifying that it means a nonterminal symbol (indexed with $\boldsymbol{v}$). The dimension $\dim([\![\boldsymbol{v}]\!])$ is $|\boldsymbol{v}|$. The set of initial symbols is

$$I = \{\, [\![\langle w\rangle]\!] \mid \langle w\rangle \in K \text{ and } w \in L_* \,\},$$

where every element of $I$ is of dimension 1. The set $F$ of functions consists of all the $\lambda$-free and non-permuting functions that appear in the definition of $P$. The rules of $P$ are divided into the following two types:

- (Type I) $[\![\boldsymbol{v}]\!] \to f([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$, if $0 \leq n \leq r$, $\boldsymbol{v}, \boldsymbol{v}_1, \dots, \boldsymbol{v}_n \in K$ and $\boldsymbol{v} = f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$ for $f$ $\lambda$-free and non-permuting;
- (Type II) $[\![\boldsymbol{u}]\!] \to [\![\boldsymbol{v}]\!]$, if $L_*/\boldsymbol{u} \cap X = L_*/\boldsymbol{v} \cap X$ and $\boldsymbol{u}, \boldsymbol{v} \in K$,

where rules of the form $[\![\boldsymbol{v}]\!] \to [\![\boldsymbol{v}]\!]$ are of Type I and Type II at the same time, but they are anyway superfluous. Obviously rules of Type II form an equivalence relation in $V$. One can merge nonterminals in each equivalence class to compact the grammar, but the results are slightly easier to present in this non-compact form.

We want each nonterminal symbol $[\![\boldsymbol{v}]\!]$ to derive $\boldsymbol{v}$. Here the construction of $I$ appears to be trivial: initial symbols derive elements of $K$ if and only if they are in the language $L_*$. This property is realized by the rules of Type I. For example, for $p = r = 2$ and $K = \mathcal{S}^{\leq 2}(\{ab\})$, one has the following rules $\pi_1, \dots, \pi_5$ of Type I that have $[\![\langle a, b\rangle]\!]$ on their left hand side:

$$\begin{aligned}
&\pi_1 : [\![\langle a, b\rangle]\!] \to \langle a, b\rangle,\\
&\pi_2 : [\![\langle a, b\rangle]\!] \to f_a([\![\langle b\rangle]\!]) \quad \text{with} \quad f_a(\langle z\rangle) = \langle a, z\rangle,\\
&\pi_3 : [\![\langle a, b\rangle]\!] \to f_b([\![\langle a\rangle]\!]) \quad \text{with} \quad f_b(\langle z\rangle) = \langle z, b\rangle,\\
&\pi_4 : [\![\langle a, b\rangle]\!] \to g([\![\langle a\rangle]\!], [\![\langle b\rangle]\!]) \;\text{ with }\; g(\langle z_1\rangle, \langle z_2\rangle) = \langle z_1, z_2\rangle,\\
&\pi_5 : [\![\langle a, b\rangle]\!] \to [\![\langle a, b\rangle]\!],
\end{aligned}$$

where $\pi_1$ indeed derives $\langle a, b\rangle$, while $\pi_5$ is superfluous. Instead of deriving $\langle a, b\rangle$ directly by $\pi_1$, one can derive it by two steps with $\pi_3$ and $\pi_6 : [\![\langle a\rangle]\!] \to \langle a\rangle$ (or $\pi_2$ and $\pi_7 : [\![\langle b\rangle]\!] \to \langle b\rangle$), or by three steps by $\pi_4$, $\pi_6$ and $\pi_7$. One may regard application of rules of Type I as a decomposition of the multiword that appears on its left hand side. It is easy to see that there are finitely many rules of Type I, because $K$ is finite and nonterminals on the right hand side of a rule are all $\lambda$-free sub-multiwords of that on the left hand side. If the grammar had only rules of Type I, then it should derive all and only elements of $I$.

The intuition behind rules of Type II is explained as follows. If $L_*/\boldsymbol{u} = L_*/\boldsymbol{v}$, then $L_*$ is closed under exchanging occurrences of $\boldsymbol{v}$ and $\boldsymbol{u}$ in any strings, and such an exchange is realized by the two symmetric rules $[\![\boldsymbol{u}]\!] \to [\![\boldsymbol{v}]\!]$ and $[\![\boldsymbol{v}]\!] \to [\![\boldsymbol{u}]\!]$ of Type II. The algorithm cannot check whether $L_*/\boldsymbol{u} = L_*/\boldsymbol{v}$ in finitely many steps, but it can approximate this relation by $L_*/\boldsymbol{u} \cap X = L_*/\boldsymbol{v} \cap X$ which we can check using MQs, because $X$ is finite. Clearly $L_*/\boldsymbol{u} = L_*/\boldsymbol{v}$ implies that $L_*/\boldsymbol{u} \cap X = L_*/\boldsymbol{v} \cap X$, but the inverse is not true. We say that a rule $[\![\boldsymbol{u}]\!] \to [\![\boldsymbol{v}]\!]$ of Type II is *incorrect (with respect to $L_*$)* if $L_*/\boldsymbol{u} \neq L_*/\boldsymbol{v}$.

In this construction of $\mathcal{G}^r(K, X, L_*)$, the initial symbols are determined by $K$ and $L_*$ and the rules of Type I are constructed solely by $K$, while $X$ is used only for determining rules of Type II. Our algorithm monotonically increases $K$ which will monotonically increase the set of rules, and monotonically increases $X$ which will decrease the set of rules of Type II.

## 3.3 Observation Tables

The process of computing rules of Type II can be handled by a collection of matrices, called *observation tables.* For each dimension $m \leq p$, we have an observation table $T_m$. Let $K_m$ and $X_m$ be the sets of $m$-words from $K$ and $m$-contexts from $X$, respectively. The rows of the table $T_m$ are indexed with the elements of $K_m$ and the columns are indexed with the elements of $X_m$. For each pair $\boldsymbol{u}, \boldsymbol{v} \in K_m$, to compare the sets $L_*/\boldsymbol{u} \cap X$ and $L_*/\boldsymbol{v} \cap X$, one needs to know whether $\mathbf{x} \odot \boldsymbol{u} \in L_*$ or not for all of $\mathbf{x} \in X_m$. The membership of $\mathbf{x} \odot \boldsymbol{u}$ is recorded in the corresponding entry of the observation table with the aid of an MQ. By comparing the entries of the rows indexed with $\boldsymbol{u}$ and $\boldsymbol{v}$, one can determine whether the grammar should have the rule $[\![\boldsymbol{u}]\!] \to [\![\boldsymbol{v}]\!]$.

*Example 3.* Let $p = 2$, $r = 1$ and

$$L_* = \{\, a^m b^n c^m d^n \mid m + n \geq 1 \,\} \in \mathbb{CL}(2, 1).$$

Indeed $L_*$ is generated by the 2-CMCFG $G_* \in \mathbb{CG}(2, 1)$ whose rules are

$$\begin{gathered} S \to f(E) \text{ with } f(\langle z_1, z_2 \rangle) = \langle z_1 z_2 \rangle, \\ E \to g_a(E) \text{ with } g_a(\langle z_1, z_2 \rangle) = \langle a z_1, c z_2 \rangle, \quad E \to \langle a, c \rangle, \\ E \to g_b(E) \text{ with } g_b(\langle z_1, z_2 \rangle) = \langle z_1 b, z_2 d \rangle, \quad E \to \langle b, d \rangle \end{gathered}$$

and whose initial symbol is $S$ only.

Let $\hat{G} = \mathcal{G}^1(K, X, L_*)$ for

$$\begin{aligned} K &= \{\, \langle abcd \rangle,\ \langle a, c \rangle,\ \langle b, d \rangle,\ \langle ab, cd \rangle,\ \langle aab, ccd \rangle \,\}, \\ X &= \{\, \square,\ a\square bc\square d,\ ab\square cd\square \,\}. \end{aligned}$$

We have the following rules of Type I:

$$\pi_1 : [\![\langle abcd\rangle]\!] \to \langle abcd\rangle, \quad \pi_2 : [\![\langle abcd\rangle]\!] \to f([\![\langle ab, cd\rangle]\!]),$$
$$\pi_3 : [\![\langle abcd\rangle]\!] \to f \circ g_b([\![\langle a, c\rangle]\!]), \quad \pi_4 : [\![\langle abcd\rangle]\!] \to f \circ g_a([\![\langle b, d\rangle]\!]),$$
$$\pi_5 : [\![\langle a, c\rangle]\!] \to \langle a, c\rangle, \quad \pi_6 : [\![\langle b, d\rangle]\!] \to \langle b, d\rangle, \quad \pi_7 : [\![\langle ab, cd\rangle]\!] \to \langle ab, cd\rangle,$$
$$\pi_8 : [\![\langle ab, cd\rangle]\!] \to g_b([\![\langle a, c\rangle]\!]), \quad \pi_9 : [\![\langle ab, cd\rangle]\!] \to g_a([\![\langle b, d\rangle]\!]),$$
$$\pi_{10} : [\![\langle aab, ccd\rangle]\!] \to \langle aab, ccd\rangle, \quad \pi_{11} : [\![\langle aab, ccd\rangle]\!] \to g_a([\![\langle ab, cd\rangle]\!]),$$
$$\pi_{12} : [\![\langle aab, ccd\rangle]\!] \to g_{aa}([\![\langle b, d\rangle]\!]) \quad \text{with} \quad g_{aa}(\langle z_1, z_2\rangle) = \langle aaz_1, ccz_2\rangle,$$
$$\pi_{13} : [\![\langle aab, ccd\rangle]\!] \to h_1([\![\langle a, c\rangle]\!]) \quad \text{with} \quad h_1(\langle z_1, z_2\rangle) = \langle z_1ab, z_2cd\rangle,$$
$$\pi_{14} : [\![\langle aab, ccd\rangle]\!] \to h_2([\![\langle a, c\rangle]\!]) \quad \text{with} \quad h_2(\langle z_1, z_2\rangle) = \langle z_1ab, cz_2d\rangle,$$
$$\pi_{15} : [\![\langle aab, ccd\rangle]\!] \to h_3([\![\langle a, c\rangle]\!]) \quad \text{with} \quad h_3(\langle z_1, z_2\rangle) = \langle az_1b, z_2cd\rangle,$$
$$\pi_{16} : [\![\langle aab, ccd\rangle]\!] \to h_4([\![\langle a, c\rangle]\!]) \quad \text{with} \quad h_4(\langle z_1, z_2\rangle) = \langle az_1b, cz_2d\rangle,$$

where superfluous rules of the form $[\![\boldsymbol{v}]\!] \to [\![\boldsymbol{v}]\!]$ are suppressed. On the other hand we have the following observation tables:

| $T_1$ | $\square$ |
|---|---|
| $\langle abcd\rangle$ | 1 |

| $T_2$ | $a\square bc\square d$ | $ab\square cd\square$ |
|---|---|---|
| $\langle a, c\rangle$ | 1 | 0 |
| $\langle b, d\rangle$ | 1 | 1 |
| $\langle ab, cd\rangle$ | 1 | 0 |
| $\langle aab, ccd\rangle$ | 1 | 0 |

Thus the three nonterminals $[\![\langle a, c\rangle]\!]$, $[\![\langle ab, cd\rangle]\!]$ and $[\![\langle aab, ccd\rangle]\!]$ can be identified thanks to the corresponding rules of Type II.

$$\rho_1\colon [\![\langle a, c\rangle]\!] \to [\![\langle ab, cd\rangle]\!],\ \rho_2\colon [\![\langle a, c\rangle]\!] \to [\![\langle aab, ccd\rangle]\!],\ \rho_3\colon [\![\langle ab, cd\rangle]\!] \to [\![\langle aab, ccd\rangle]\!],$$
$$\rho_4\colon [\![\langle ab, cd\rangle]\!] \to [\![\langle a, c\rangle]\!],\ \rho_5\colon [\![\langle aab, ccd\rangle]\!] \to [\![\langle a, c\rangle]\!],\ \rho_6\colon [\![\langle aab, ccd\rangle]\!] \to [\![\langle ab, cd\rangle]\!].$$

The unique initial symbol of $\hat{G}$ is $[\![\langle abcd\rangle]\!]$.

Let us see some derivations of $\hat{G}$. Derivation trees of the form

$$\overbrace{\rho_3(\pi_{11}(\dots(\rho_3(\pi_{11}}^{(m-1)\text{-times}}(\rho_4(\pi_5))))\dots))$$

give us $\langle a^m, c^m\rangle \in \mathcal{L}([\![\langle ab, cd\rangle]\!])$ for all $m \geq 1$. Similarly

$$\overbrace{\pi_8(\rho_1(\dots(\pi_8(\rho_1}^{n\text{-times}}\overbrace{(\rho_3(\pi_{11}(\dots(\rho_3(\pi_{11}}^{(m-1)\text{-times}}(\rho_4(\pi_5))))\dots)))))\dots))$$

give us $\langle a^m b^n, c^m d^n\rangle \in \mathcal{L}([\![\langle ab, cd\rangle]\!])$ for all $n \geq 0$. Finally by applying $\pi_2$, we see $a^m b^n c^m d^n \in \mathcal{L}(\hat{G})$.

However the rules $\rho_1, \rho_2, \rho_4, \rho_5$ of Type II, which involve $[\![\langle a, c\rangle]\!]$, are all incorrect, because $\square ab\square cd \in L_*/\langle a, c\rangle - L_*/\langle ab, cd\rangle = L_*/\langle a, c\rangle - L_*/\langle aab, ccd\rangle$. In fact the derivation tree $\pi_2(\rho_3(\pi_{13}(\rho_1(\pi_7))))$, for instance, yields $ababcdcd \in \mathcal{L}(\hat{G}) - L_*$.

**Lemma 4.** *One can compute $\hat{G} = \mathcal{G}^r(K, X, L_*)$ in polynomial time in $\|K\|$ and $\|X\|$.*

*Proof.* We first estimate the number of rules of Type I that have a fixed $[\![\boldsymbol{v}]\!]$ on the left hand side, which are of the form $[\![\boldsymbol{v}]\!] \to f([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$. Roughly speaking, this is the number of ways to decompose $\boldsymbol{v}$ into sub-multiwords $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ with the aid of a linear regular function $f$. Once one has fixed where the occurrence of each component from $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$ starts and ends in $\boldsymbol{v}$, the function $f$ is uniquely determined. We have at most $pr$ components in $\boldsymbol{v}_1, \dots, \boldsymbol{v}_n$, hence the number of ways to determine such starting and ending points are at most $\|\boldsymbol{v}\|^{2pr}$. Thus, the number of rules of Type I is at most $O(|K|\ell^{2pr})$ where $\ell$ is the maximal size of elements of $K$. Clearly the description size of each rule is at most $O(\ell)$.

One can construct the observation tables by at most $|K||X|$ MQs. Then one can determine initial symbols and rules of Type II in polynomial time. $\square$

The next subsection discusses how our learner determines $K$ and $X$.

### 3.4 Undergeneralization

The problems that we have to deal with are undergeneralization and overgeneralization. We first show when the data are sufficient to avoid undergeneralization.

**Lemma 5.** *Let $G_* \in \mathbb{CG}(p, r)$ be such that $\mathcal{L}(G_*) = L_*$. If a finite set $D \subseteq L_*$ is such that every rule of $G_*$ occurs in a derivation tree for some $w \in D$, then for any $X$ such that $\Box \in X$, we have $L_* \subseteq \mathcal{L}(\hat{G})$ where $\hat{G} = \mathcal{G}^r(\mathcal{S}^{\leq p}(D), X, L_*)$.*

*Proof.* Let $G_* = \langle \Sigma, V_*, F_*, P_*, I_* \rangle$ and $K = \mathcal{S}^{\leq p}(D)$. By induction we show that if $\boldsymbol{w} \in \mathcal{L}(G_*, A)$, then $\boldsymbol{w} \in \mathcal{L}(\hat{G}, [\![\boldsymbol{v}]\!])$ for some $\boldsymbol{v} \in \mathcal{L}(G_*, A) \cap K$. In particular when $A \in I_*$, $[\![\boldsymbol{v}]\!]$ will be an initial symbol of $\hat{G}$, so this proves the lemma.

Suppose that $\pi(t_1, \dots, t_n)$ is an $A$-derivation tree for $\boldsymbol{w}$ of $G_*$ where $\pi$ is of the form $A \to f(B_1, \dots, B_n)$ and $t_i$ is a $B_i$-derivation tree for $\boldsymbol{w}_i$ for all $i = 1, \dots, n$. By induction hypothesis, there are $\boldsymbol{u}_i \in \mathcal{L}(G_*, B_i) \cap K$ such that $\boldsymbol{w}_i \in \mathcal{L}(\hat{G}, [\![\boldsymbol{u}_i]\!])$ for all $i$. On the other hand, by the assumption, we have $w \in D$ that is derived by using $\pi$, which can be represented as

$$w = \mathbf{x} \odot \boldsymbol{v} = \mathbf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$$

where $\boldsymbol{v}_i \in \mathcal{L}(G_*, B_i) \cap K$ for all $i = 1, \dots, n$ and $f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n) \in \mathcal{L}(G_*, A) \cap K$. Let $\boldsymbol{v} = f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$. Then $\hat{G}$ has the rule

$$[\![\boldsymbol{v}]\!] \to f([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$$

of Type I and moreover rules $[\![\boldsymbol{v}_i]\!] \to [\![\boldsymbol{u}_i]\!]$ of Type II for all $i = 1, \dots, n$ whatever $X$ is, since $G_*$ is a CMCFG. By applying those rules of Types I and II to $\boldsymbol{w}_i \in \mathcal{L}(\hat{G}, [\![\boldsymbol{u}_i]\!])$ for $i = 1, \dots, n$, we obtain $\boldsymbol{w} \in \mathcal{L}(\hat{G}, [\![\boldsymbol{v}]\!])$ with $\boldsymbol{v} \in \mathcal{L}(G_*, A) \cap K$. $\square$

When our algorithm gets a positive counter-example $w \in L_* - \mathcal{L}(\hat{G})$, it adds all multiwords in $\mathcal{S}^{\leq p}(\{w\})$ to $K$.

## 3.5 Overgeneralization

Overgeneralization is a little more difficult to treat. Because overgeneralization is caused by incorrect rules of Type II, we must add appropriate multicontexts to $X$ in order to remove them.

Suppose that our conjecture $\hat{G}$ is a correct CMCFG for the target $L_*$. Then it must satisfy that

– for any $[\![\boldsymbol{v}]\!] \in V$ and any $\boldsymbol{u} \in \mathcal{L}(\hat{G}, [\![\boldsymbol{v}]\!])$, it holds that $L_*/\boldsymbol{u} = L_*/\boldsymbol{v}$.

When we get a negative counter-example $w$ from the teacher, this means that we have $\langle w \rangle \in \mathcal{L}(\hat{G}, [\![\langle v \rangle]\!])$ for some initial symbol $[\![\langle v \rangle]\!]$, but $\square \in L_*/\langle v \rangle - L_*/\langle w \rangle$. The triple $([\![\langle v \rangle]\!], \langle w \rangle, \square)$ is a witness for the fact that $\hat{G}$ does not satisfy the above desired property. It is not hard to see by Lemma 3 that in such a case $\hat{G}$ must have an incorrect rule of Type II which is used for deriving $w$. In order to find such an incorrect rule, we first parse the string $w$ with $\hat{G}$ and get a derivation tree $t$ for $w$. We then look for an incorrect rule in $t$ that causes this violation by recursively searching $t$ in a topdown manner as described below, where the initial value of $\mathbf{x}$ is $\square$.

Suppose that we have a pair $(t, \mathbf{x})$ such that $t$ is a $[\![\boldsymbol{v}]\!]$-derivation tree for $\boldsymbol{u}$ and $\mathbf{x} \in L_*/\boldsymbol{v} - L_*/\boldsymbol{u}$. We have two cases.

CASE 1. Suppose that $t = \pi(t_1, \dots, t_n)$ for some rule $\pi$ of Type I. Then, there are $[\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!] \in V$, $\boldsymbol{u}_1, \dots, \boldsymbol{u}_n \in (\Sigma^+)^{\langle * \rangle}$ and $f$ such that

– $\pi$ is of the form $[\![\boldsymbol{v}]\!] \to f([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$, i.e., $\boldsymbol{v} = f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n)$,
– $\boldsymbol{u} = f(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n)$,
– $t_i$ is a $[\![\boldsymbol{v}_i]\!]$-derivation tree for $\boldsymbol{u}_i$ for $i = 1, \dots, n$.

By the assumption we have

$$\mathbf{x} \odot \boldsymbol{v} = \mathbf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_n) \in L_*,$$
$$\mathbf{x} \odot \boldsymbol{u} = \mathbf{x} \odot f(\boldsymbol{u}_1, \dots, \boldsymbol{u}_n) \notin L_*.$$

This means that $n$ cannot be 0. One can find $k$ such that

$$\mathbf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_{k-1}, \boldsymbol{v}_k, \boldsymbol{u}_{k+1}, \dots, \boldsymbol{u}_n) \in L_*,$$
$$\mathbf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_{k-1}, \boldsymbol{u}_k, \boldsymbol{u}_{k+1}, \dots, \boldsymbol{u}_n) \notin L_*.$$

That is, for

$$\mathbf{x}' = \mathbf{x} \odot f(\boldsymbol{v}_1, \dots, \boldsymbol{v}_{k-1}, \square^{\langle |\boldsymbol{v}_k| \rangle}, \boldsymbol{u}_{k+1}, \dots, \boldsymbol{u}_n),$$

we have $\mathbf{x}' \in L/\boldsymbol{v}_k - L/\boldsymbol{u}_k$. We then recurse with $(t_k, \mathbf{x}')$.

CASE 2. Suppose that $t = \pi(t')$ where $\pi : [\![\boldsymbol{v}]\!] \to [\![\boldsymbol{v}']\!]$ is a rule of Type II. Then $t'$ is a $[\![\boldsymbol{v}']\!]$-derivation tree for $\boldsymbol{u}$. If $\mathbf{x} \odot \boldsymbol{v}' \notin L_*$, this means that $[\![\boldsymbol{v}]\!] \to [\![\boldsymbol{v}']\!]$ is an incorrect rule, because $\mathbf{x} \in L_*/\boldsymbol{v} - L_*/\boldsymbol{v}'$. We then add $\mathbf{x}$ to $X$ for removing the incorrect rules $[\![\boldsymbol{v}]\!] \to [\![\boldsymbol{v}']\!]$ and $[\![\boldsymbol{v}']\!] \to [\![\boldsymbol{v}]\!]$. Otherwise, we know that $\mathbf{x} \in L_*/\boldsymbol{v}' - L_*/\boldsymbol{u}$. We recurse with $(t', \mathbf{x})$.

We use FindContext($\hat{G}, w$) to refer to this procedure for computing a multicontext that witnesses incorrect rules from a negative counter-example $w$. The size of the output $\mathbf{x}$ is bounded by $|w|\ell$ for $\ell = \max\{\, \|\boldsymbol{v}\| \mid \boldsymbol{v} \in K \,\}$, because $\mathbf{x}$ is obtained from $w$ by replacing some occurrences of nonempty sub-multiwords $\boldsymbol{u}_i$ by $\boldsymbol{v}_i \in K$ and an occurrence of a sub-multiword by a hole $\Box$. The procedure FindContext contains parsing as a subroutine. We can use any of polynomial-time parsing algorithms for MCFGs proposed so far, e.g., [13, 19].

**Lemma 6.** FindContext($\hat{G}, w$) *runs in polynomial time in* $|w|$ *and* $\|\hat{G}\|$.

## 3.6 Algorithm

We are now ready to describe the overall structure of our algorithm. The pseudocode is presented in Algorithm 1.

**Algorithm 1** Learn $\mathbb{CL}(p, r)$

```
let $K := \varnothing$; $X := \{\Box\}$; $\hat{G} = \mathcal{G}^r(K, X, L_*)$;
while the teacher does not answer "YES" to the EQ on $\hat{G}$ do
  let $w$ be the counter-example from the teacher;
  if $w \in L_* - \mathcal{L}(\hat{G})$ then
    let $K := K \cup \mathcal{S}^{\le p}(\{w\})$;
  else
    let $X := X \cup \{\text{FindContext}(\hat{G}, w)\}$;
  end if
  let $\hat{G} = \mathcal{G}^r(K, X, L_*)$;
end while
output $\hat{G}$;
```

Let $G_* = \langle \Sigma, V_*, F_*, P_*, I_* \rangle \in \mathbb{CG}(p, r)$ represent the learning target $L_*$.

**Lemma 7.** *The algorithm receives a positive counter-example at most $|P_*|$ times. The cardinality of $K$ is bounded by $O(|P_*|\ell^{2p})$, where $\ell$ is the length of a longest positive counter-example given so far.*

*Proof.* The proof of Lemma 5 in fact claims that any of the rules of $G_*$ that are used for deriving positive counter-examples can be simulated by $\hat{G}$. That is, whenever the learner gets a positive counter-example $w$ from the teacher, there is a rule of $G_*$ that is used for deriving $w$ but not used for any of previously presented positive counter-examples. Therefore the learner receives a positive counter-example at most $|P_*|$ times.

It is easy to see that $|\mathcal{S}^{\le p}(\{w\})| \in O(|w|^{2p})$. Hence $|K| \in O(|P_*|\ell^{2p})$ where $\ell$ is the length of a longest positive counter-example given so far. $\Box$

**Lemma 8.** *The number of times that the algorithm receives a negative counter-example and the cardinality of $X$ are both bounded by $O(|P_*|\ell^{2p})$.*

*Proof.* Each time the learner receives a negative counter-example, one element is added to $X$ and some incorrect rules of Type II are removed. That is, the number of equivalence classes induced by rules of Type II is increased. Because there can be at most $|K|$ equivalence classes in $V$, the learner expands $X$ at most $|K|$ times, that is, $|X| \in O(|P_*|\ell^{2p})$ by Lemma 7. □

**Theorem 2.** *Our algorithm outputs a grammar representing the target language $L_*$ and halts in polynomial time in $\|G_*\|$ and $\ell$ where $G_* \in \mathbb{CG}(p, r)$ is a grammar representing $L_*$ and $\ell$ is the length of a longest counter-example from the teacher.*

*Proof.* By Lemmas 4 and 6, each time the learner receives a counter-example, it computes the next conjecture in polynomial time in $\|G_*\|$ and $\ell$. By Lemmas 7 and 8, it asks EQs at most polynomial times in $\|G_*\|$ and $\ell$. All in all, it runs in polynomial time in $\|G_*\|$ and $\ell$. The correctness of the output of the algorithm is guaranteed by the teacher. □

### 3.7 Slight Enhancement

Let us say that $L \in \mathbb{L}(p, r)$ is *almost p-congruential* if $\$L\$ \in \mathbb{CL}(p, r)$ where \$ is a new marker not in $\Sigma$. While every language in $\mathbb{CL}(p, r)$ is almost $p$-congruential, the converse does not hold. An example in the difference is $L_{\text{copy}} = \{\, w\overline{w} \in \{a, b, \bar{a}, \bar{b}\}^* \mid w \in \{a, b\}^* \,\}$, where $\bar{\cdot}$ denotes the homomorphism that maps $a, b$ to $\bar{a}, \bar{b}$, respectively. One can easily modify our learning algorithm so that it learns all almost $p$-congruential languages in $\mathbb{L}(p, r)$ just by assuming the marker \$ on the head and the tail on each counter-example.

## 4 Conclusion

This work is based on a combination of the MAT algorithm for learning congruential CFGs presented in [17] and the algorithm for learning some sorts of MCFGs from positive data and MQs in [20].

The conjecture $\hat{G}$ of our algorithm is not always consistent with the observation tables in the sense that it might be the case $\mathcal{L}(\hat{G}) \cap (X \odot K) \neq L_* \cap (X \odot K)$. For $\mathbf{x} \in X$ and $\boldsymbol{y} \in K$, one can regard $\mathbf{x} \odot \boldsymbol{y} \in L_* - \mathcal{L}(\hat{G})$ as a positive counter-example and $\mathbf{x} \odot \boldsymbol{y} \in \mathcal{L}(\hat{G}) - L_*$ as a negative counter-example. One can modify our algorithm so that the conjecture is always consistent by substituting this self-diagnosis method for EQs to the teacher. EQs will be used only when the learner does not find any inconsistency. Verification of the consistency can be done in polynomial time, however we do not employ this idea in our algorithm, because each time the learner extracts a counter-example from the observation tables, they will recursively be expanded and this might cause exponential-time computation, when the size of a counter-example computed by the learner is not considered to be a parameter of the input size.

Lemma 3 offers another method for finding incorrect rules in some cases without getting a negative counter-example from the teacher. That is, if there are rules of Type I $[\![\boldsymbol{u}]\!] \to f([\![\boldsymbol{u}_1]\!], \dots, [\![\boldsymbol{u}_n]\!])$ and $[\![\boldsymbol{v}]\!] \to f([\![\boldsymbol{v}_1]\!], \dots, [\![\boldsymbol{v}_n]\!])$ such that $L_*/\boldsymbol{u}_i \cap X = L_*/\boldsymbol{v}_i \cap X$ for all $i$ and $L_*/\boldsymbol{u} \cap X \neq L_*/\boldsymbol{v} \cap X$, then one knows that $[\![\boldsymbol{u}_k]\!] \to [\![\boldsymbol{v}_k]\!]$ is an incorrect rule for some $k$. Yet we do not take this idea into our algorithm for the same reason discussed in the previous paragraph.

In Clark et al. [21] the approach uses representational primitives that correspond not to congruence classes but to elements of the form:

$$\{\, v \mid L/v \supseteq L/u \,\} \tag{1}$$

An advantage of using congruence classes is that any element of the class will be as good as any other element; if we use the idea of (1), we cannot be sure that we will get the right elements, and it is difficult to get a pure MAT result.

Distributional lattice grammars (DLGs) are another interesting development in the field of rich language classes that are also learnable [22]. While these models can represent some non-context-free languages, it is not yet clear whether they are rich enough to account for the sorts of cross-serial dependency that occur in natural languages; they are basically context free grammars, with an additional operation, somewhat analogous to that of Conjunctive Grammars [23].

We suggest two directions for future research: a PAC-learning result along the lines of the PAC result for NTS languages in [24] seems like a natural extension, since MCFGs have a natural stochastic variant. Secondly, combining DLGs and MCFGs might be possible – we would need to replace the single concatenation operation in DLGs with a more complex range of operations that reflect the variety of ways that tuples of strings can be combined.

We also note that given the well-known relationship between synchronous CFGs and MCFGs [25] this algorithm gives as a special case the first learning algorithm for learning synchronous context free grammars, and therefore of syntax-directed translation schemes. Given the current interest in syntax-based statistical machine translation, we intend to explore further this particular class of algorithms.

## Acknowledgement

The authors are very grateful to Anna Kasprzik and the anonymous reviewers for valuable comments and suggestions on a draft of this paper.

## References

1. Joshi, A.K.: Tree adjoining grammars: how much context-sensitivity is required to provide reasonable structural descriptions? In Dowty, D.R., Karttunen, L., Zwicky, A., eds.: Natural Language Parsing. Cambridge University Press, Cambridge, MA (1985) 206–250
2. Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: Proceedings of the 25th annual meeting of Association for Computational Linguistics, Stanford (1987) 104–111
3. Joshi, A.K., Levy, L.S., Takahashi, M.: Tree adjunct grammars. Journal of Computer and System Sciences **10**(1) (1975) 136–163

4. Joshi, A.K.: An introduction to tree adjoining grammars. In Manaster-Ramer, A., ed.: Mathematics of Languge. John Benjamins (1987)
5. Stabler, E.P.: Derivational minimalism. In Retoré, C., ed.: LACL. Volume 1328 of Lecture Notes in Computer Science., Springer (1996) 68–95
6. Engelfriet, J., Heyker, L.: The string generating power of context-free hypergraph grammars. Journal of Computer and System Sciences **43**(2) (1991) 328–360
7. Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. Journal of Machine Learning Research **8** (2007) 1725–1745
8. Huybrechts, R.A.C.: The weak inadequacy of context-free phrase structure grammars. In de Haan, G., Trommelen, M., Zonneveld, W., eds.: Van Periferie naar Kern. Foris, Dordrecht, Holland (1984)
9. Shieber, S.M.: Evidence against the context-freeness of natural language. Linguistics and Philosophy **8** (1985) 333–343
10. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2) (1987) 87–106
11. Angluin, D., Kharitonov, M.: When won't membership queries help? Journal of Computer and System Sciences **50**(2) (1995) 336–355
12. Clark, A., Thollard, F.: PAC-learnability of probabilistic deterministic finite state automata. Journal of Machine Learning Research **5** (May 2004) 473–497
13. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science **88**(2) (1991) 191–229
14. Kracht, M. In: The Mathematics of Language. Volume 63 of Studies in Generative Grammar. Mouton de Gruyter (2003) 408–409
15. Rambow, O., Satta, G.: Independent parallelism in finite copying parallel rewriting systems. Theoretical Computer Science **223**(1-2) (1999) 87–120
16. Kaji, Y., Nakanishi, R., Seki, H., Kasami, T.: The universal recognition problems for parallel multiple context-free grammars and for their subclasses. IEICE Transaction on Information and Systems **E75-D**(7) (1992) 499–508
17. Clark, A.: Distributional learning of some context-free languages with a minimally adequate teacher. In: Proceedings of the ICGI, Valencia, Spain (September 2010)
18. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2) (1987) 87–106
19. Kanazawa, M.: A prefix-correct earley recognizer for multiple context-free grammars. In: proceedings of the Ninth International Workshop on Tree Adjoining Grammars and Related Formalisms. (2008) 49–56
20. Yoshinaka, R.: Polynomial-time identification of multiple context-free languages from positive data and membership queries. In Sempere, J.M., García, P., eds.: ICGI. Volume 6339 of Lecture Notes in Computer Science., Springer (2010) 230–244
21. Clark, A., Eyraud, R., Habrard, A.: A polynomial algorithm for the inference of context free languages. In Clark, A., Coste, F., Miclet, L., eds.: ICGI. Volume 5278 of Lecture Notes in Computer Science., Springer (2008) 29–42
22. Clark, A.: A learnable representation for syntax using residuated lattices. In: Proceedings of the 14th Conference on Formal Grammar, Bordeaux, France (2009)
23. Okhotin, A.: Conjunctive grammars. Journal of Automata, Languages and Combinatorics **6**(4) (2001) 519–535
24. Clark, A.: PAC-learning unambiguous NTS languages. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., Tomita, E., eds.: ICGI. Volume 4201 of Lecture Notes in Computer Science., Springer (2006) 59–71
25. Melamed, I.D.: Multitext grammars and synchronous parsers. In: Proceedings of NAACL/HLT. (2003) 79–86