# Learning Context-Free Grammars from Positive Data and Membership Queries

Makoto Kanazawa

Faculty of Science and Engineering, Hosei University, Japan
kanazawa@hosei.ac.jp
https://makotokanazawa.ws.hosei.ac.jp

**Abstract.** I review some recent results on learning subclasses of the context-free grammars from positive data and membership queries. I motivate the relevant learning algorithms through comparison with a similar and arguably reasonable learning algorithm for regular languages.

**Keywords:** Grammatical inference · Membership queries · Context-free grammars · Extended regular closure

## 1 Introduction

This paper concerns algorithmic learning of context-free languages. The "learning paradigm" we use is *polynomial-time identification in the limit from positive data and membership queries.* The learner receives an infinite stream of positive examples enumerating the target language, and each time it receives a new positive example, it makes polynomially many queries to the oracle for the target language before outputting a hypothesis. The goal is to converge to a correct grammar for the target language. The availability of the membership oracle makes it possible to learn some interesting subclasses of the context-free languages that properly include the regular languages. This paper presents a variant of previously proposed learning algorithms for three such classes [9,5,6,7].

A key difficulty in learning context-free, as opposed to regular, languages lies in the relationship between the string sets corresponding to the nonterminals of a context-free grammar and the generated language. In the case of a regular language, states of a minimal DFA for the language correspond to its *left quotients.*[1] A left quotient of a language $L$ is a language of the form $u\backslash L = \{\, x \mid ux \in L \,\}$, where $u$ is some string. In order to determine whether a string $x$ belongs to $u\backslash L$, the learner can just ask the membership oracle whether $ux$ belongs to $L$. Furthermore, when $L$ is regular, there are only finitely many left quotients of $L$, and this makes it possible to identify the set of left quotients of $L$ in the limit.

In the case of a context-free grammar $G$, the relationship between the set of strings derived from a nonterminal $A$ of $G$ and the language $L = L(G)$ of $G$ is much less straightforward. Unless $A$ is useless, there is a pair of terminal strings

[1] In this paper, by a "minimal DFA" for a regular language, we mean one with no dead state.

$(u, v)$ such that $S \Rightarrow_G^* u\,A\,v$, so the set $L_G(A) = \{\, x \mid A \Rightarrow_G^* x \,\}$ must be a subset of $u\backslash L/v = \{\, x \mid uxv \in L \,\}$. (A set of this latter form is called a *quotient* of $L$.) In general, $L_G(A)$ may be a proper subset of $\bigcap\{\, u\backslash L/v \mid S \Rightarrow_G^* uAv \,\}$, and it is not clear whether there is anything further that can be said in general about the relationship between $L_G(A)$ and the quotients of $L$.

The kind of learning algorithm we look at in this paper simply assumes that the string set associated with each nonterminal of the target grammar $G_*$ can be expressed as the result of applying certain operations to quotients of $L_* = L(G_*)$. We consider three successively larger sets of operations that may be used in these expressions: (i) the set consisting of intersection only, (ii) the set of Boolean operations, and (iii) the set consisting of Boolean and regular operations. With the choice (i), the string sets associated with the nonterminals are in the *intersection closure* of $\mathrm{Quot}(L_*) = \{\, u\backslash L_*/v \mid (u, v) \in \Sigma^* \times \Sigma^* \,\}$, the set of quotients of $L_*$. With (ii), they are in the *Boolean closure* of $\mathrm{Quot}(L_*)$. With (iii), they are in what we call the *extended regular closure* of $\mathrm{Quot}(L_*)$. When $K$ is a language in one of these classes, the membership of a string $x$ in $K$ can be determined by making a finite number of queries of the form "$uyv \in L_*$?", where $(u, v)$ is an element of some fixed set (depending on $K$) and $y$ is a substring of $x$. As we will see, the fact that the membership problem for the set associated with a nonterminal reduces to the membership problem for the target language means that the "validity" of a production can be decided in the limit with the help of the oracle for the target language.

Before describing our learning algorithms for the three subclasses of the context-free languages (Section 3), it is perhaps instructive to look at how the class of regular languages can be learned within the same paradigm, so we start with the latter.

## 2 Regular Languages

Let us describe a learning algorithm that identifies an unknown regular language $L_*$ from positive data and membership queries. So as to facilitate comparison with the case of context-free languages, we assume that the learner outputs right-linear context-free grammars. A context-free grammar $G = (N, \Sigma, P, S)$, where $N$ is the set of nonterminals, $\Sigma$ is the terminal alphabet, $P$ is the set of productions, and $S$ is the start symbol, is said to be *right-linear* if each production in $P$ is either of the form $A \rightarrow a\,B$ or of the form $A \rightarrow \varepsilon$, where $A, B \in N$ and $a \in \Sigma$. Suppose that $G_* = (N_*, \Sigma, P_*, S)$ is the right-linear context-free grammar corresponding to the minimal DFA for $L_*$. (This means that $G_*$ has no useless nonterminal.) The sets of terminal strings derived from the nonterminals of $G_*$ are exactly the nonempty left quotients of $L_*$. For each $B \in N_*$, let $u_B$ be the length-lexicographically first string such that $u_B\backslash L_* = \{\, x \in \Sigma^* \mid B \Rightarrow_{G_*}^* x \,\}$.[2] We have $u_S = \varepsilon$, corresponding to $\varepsilon\backslash L_* = L_*$. Productions in

[2] The choice of the length-lexicographic order is not essential. Other strict total orders on $\Sigma^*$ may be used instead, provided that the empty string comes first.

$P_*$ are of one of two forms:

$$\begin{aligned} &A \to a\,B, && \text{where } a \in \Sigma \text{ and } u_A a \backslash L_* = u_B \backslash L_*\,, \\ &A \to \varepsilon, && \text{where } u_A \in L_*\,. \end{aligned}$$

(The former type of production means that there is a transition labeled $a$ from the state corresponding to $A$ to the state corresponding to $B$ in the minimal DFA, and the latter type of production means that $A$ corresponds to a final state.) The learner's task is (i) to identify the set $Q_* = \{\, u_B \mid B \in N_* \,\}$, and (ii) to determine, for each $u, v \in Q_*$ and $a \in \Sigma$, whether $ua \backslash L_* = v \backslash L_*$.

For $K \subseteq \Sigma^*$, let

$$\begin{aligned} \mathrm{Pref}(K) &= \{\, u \in \Sigma^* \mid uv \in K \text{ for some } v \in \Sigma^* \,\}, \\ \mathrm{Suff}(K) &= \{\, v \in \Sigma^* \mid uv \in K \text{ for some } u \in \Sigma^* \,\}. \end{aligned}$$

One reasonable strategy for the learner is to work under the assumption that the available positive data $T$ is large enough that $Q_* \subseteq \mathrm{Pref}(T)$ and for each pair of distinct nonempty left quotients of $L_*$, a string in their symmetric difference occurs in $(\{\varepsilon\} \cup \Sigma)\,\mathrm{Suff}(T)$. (The assumption will eventually be true.) Let $\prec$ be the length-lexicographic strict total order on $\Sigma^*$. For $J, E \subseteq \Sigma^*$, define

$$\begin{aligned} Q(J,E) = \{\, u \mid{}& u \in J \text{ and for every } v \in J, \\ & \text{if } v \prec u, \text{ then } (\{\varepsilon\} \cup \Sigma)E \cap (v \backslash L_*) \neq (\{\varepsilon\} \cup \Sigma)E \cap (u \backslash L_*) \,\}. \end{aligned}$$

Then $Q(\mathrm{Pref}(T), \mathrm{Suff}(T))$ is the set of nonterminals of the grammar the learner hypothesizes. When we use a string $u$ as a nonterminal in a grammar, we write $\langle\!\langle u \rangle\!\rangle$ instead of just $u$ to avoid confusion. A production $\langle\!\langle u \rangle\!\rangle \to a\,\langle\!\langle v \rangle\!\rangle$ should be included in the grammar if and only if $ua \backslash L_* = v \backslash L_*$, but this cannot be decided without knowledge of the identity of $L_*$, even with the help of the oracle for $L_*$. It is again reasonable for the learner to assume that the available positive data is large enough to provide any witness to the falsity of this identity. Let

$$\begin{aligned} P(J,E) = {}& \{\, \langle\!\langle u \rangle\!\rangle \to a\,\langle\!\langle v \rangle\!\rangle \mid u, v \in J, a \in \Sigma, E \cap (ua \backslash L_*) = E \cap (v \backslash L_*) \,\} \cup \\ & \{\, \langle\!\langle u \rangle\!\rangle \to \varepsilon \mid u \in J, u \in L_* \,\}. \end{aligned} \tag{1}$$

Then $P(Q(\mathrm{Pref}(T), \mathrm{Suff}(T)), \mathrm{Suff}(T))$ is the set of productions of the hypothesized grammar.

The learning algorithm in its entirety is listed in Algorithm 1.[3]

It is not difficult to see that the output $G_i$ of Algorithm 1 is isomorphic to $G_*$ whenever the following conditions hold:[4]

(i) $Q_* \subseteq \mathrm{Pref}(T_i)$,

[3] There is an obvious connection with the work of Angluin [1,2] and many others which I will not discuss here since this algorithm is not the basis of our generalization to context-free languages.

[4] I write $X \mathbin{\triangle} Y$ for the symmetric difference of $X$ and $Y$.

```
Algorithm 1: Learner for the regular languages.
  Data: A positive presentation $t_1, t_2, \ldots$ of $L_* \subseteq \Sigma^*$; membership oracle for
        $L_*$;
  Result: A sequence of grammars $G_1, G_2, \ldots$;
  $T_0 := \varnothing$;
  for $i = 1, 2, \ldots$ do
      $T_i := T_{i-1} \cup \{t_i\}$; output $G_i := (N_i, \Sigma, P_i, \langle\!\langle \varepsilon \rangle\!\rangle)$ where
        $N_i := Q(\mathrm{Pref}(T_i), \mathrm{Suff}(T_i))$;
        $P_i := P(N_i, \mathrm{Suff}(T_i))$;
  end
```

(ii) for each $u, v \in Q_*$,

$$u \backslash L_* \neq v \backslash L_* \Longrightarrow (\{\varepsilon\} \cup \Sigma)\,\mathrm{Suff}(T_i) \cap ((u \backslash L_*) \mathbin{\triangle} (v \backslash L_*)) \neq \varnothing,$$

(iii) for each $u, v \in Q_*$ and $a \in \Sigma$,

$$ua \backslash L_* \neq v \backslash L_* \Longrightarrow \mathrm{Suff}(T_i) \cap ((ua \backslash L_*) \mathbin{\triangle} (v \backslash L_*)) \neq \varnothing.$$

Computing $N(T_i)$ requires membership queries for all elements of $\mathrm{Pref}(T_i)\,(\{\varepsilon\} \cup \Sigma)\ \mathrm{Suff}(T_i)$, while computing $P(T_i)$ requires membership queries for some subset of $\mathrm{Pref}(T_i)\,(\{\varepsilon\} \cup \Sigma)\ \mathrm{Suff}(T_i)$. The number of queries needed to compute $G_i$ is polynomial in the total lengths of the strings in $T_i$.

Algorithm 1 satisfies the following properties:

(a) It is set-driven: $G_i$ is determined uniquely by $\{t_1, \ldots, t_i\}$ (for a fixed $L_*$ but across different positive presentations $t_1, t_2, \ldots$ of $L_*$).

(b) Its conjecture is consistent with the positive data: $\{t_1, \ldots, t_i\} \subseteq L(G_i)$.[5]

(c) It updates its conjecture in polynomial time (in the total lengths of the strings in $\{t_1, \ldots, t_i\}$).

(d) There is a "characteristic sample" $D \subseteq L_*$ whose total size is polynomial in the representation size of $G_*$ such that $G_i$ is isomorphic to $G_*$ whenever $D \subseteq \{t_1, \ldots, t_i\}$.

These characteristics of Algorithm 1 obviously rest on special properties of the regular languages, and not all of them can be maintained as we move to learning algorithms for context-free languages. We keep (c), but abandon (a), (b), and (d) in favor of weaker conditions. Since a context-free language has no canonical grammar and there is no polynomial bound on the length of the shortest string generated by a context-free grammar, we cannot hope to maintain (d), but even the following weakening will not hold of our algorithms:

(d†) There is a "characteristic sample" $D \subseteq L_*$ whose *cardinality* is polynomial in the representation size of $G_*$ such that $L(G_i) = L_*$ whenever $D \subseteq T_i$.

[5] This requires a proof. Here it is crucial that we had $(\{\varepsilon\} \cup \Sigma)E$ rather than $E$ in the definition of $Q(J, E)$.

Let us illustrate the kind of change we must make with another learning algorithm for the regular languages. The algorithm will no longer be set-driven. For $J, E \subseteq \Sigma^*$, define

$$\begin{aligned} P'(J, E) = \{\, \langle\!\langle u \rangle\!\rangle \to a\, \langle\!\langle v \rangle\!\rangle \mid u, v \in J, a \in \Sigma, u \backslash L_* \supseteq a\, (E \cap (v \backslash L_*))\,\} \cup \\ \{\, \langle\!\langle u \rangle\!\rangle \to \varepsilon \mid u \in J, u \in L_*\,\}. \end{aligned} \tag{2}$$

Since $u \backslash L_* \supseteq a\,(E \cap (v \backslash L_*))$ is equivalent to $E \cap (ua \backslash L_*) \supseteq E \cap (v \backslash L_*)$, the difference between $P(J, E)$ and $P'(J, E)$ just consists in replacing equality with inclusion. The algorithm updates the set $J_i$ of prefixes of positive examples only when the positive examples received so far are incompatible with the previous conjecture. It is listed in Algorithm 2.

**Algorithm 2:** Learner for the regular languages, nondeterministic version.

```
Data: A positive presentation $t_1, t_2, \ldots$ of $L_* \subseteq \Sigma^*$; membership oracle for
      $L_*$;
Result: A sequence of grammars $G_1, G_2, \ldots$;
$T_0 := \varnothing$; $E_0 := \varnothing$; $J_0 := \varnothing$; $G_0 := (\{\langle\!\langle \varepsilon \rangle\!\rangle\}, \Sigma, \varnothing, \langle\!\langle \varepsilon \rangle\!\rangle)$;
for $i = 1, 2, \ldots$ do
    $T_i := T_{i-1} \cup \{t_i\}$; $E_i := E_{i-1} \cup \mathrm{Suff}(\{t_i\})$;
    if $T_i \subseteq L(G_{i-1})$ then
        $J_i := J_{i-1}$;
    else
        $J_i := \mathrm{Pref}(T_i)$;
    end
    $N_i := Q(J_i, E_i)$;
    $P_i := P'(N_i, E_i)$;
    output $G_i := (N_i, \Sigma, P_i, \langle\!\langle \varepsilon \rangle\!\rangle)$;
end
```

Suppose that the conditions (i), (ii) above and the following condition (iii′) hold of $T_i$:

(iii′) for each $u, v \in Q_*$ and $a \in \Sigma$,

$$u \backslash L_* \not\supseteq a(v \backslash L_*) \Longrightarrow a\,(\mathrm{Suff}(T_i) \cap (v \backslash L_*)) - (u \backslash L_*) \neq \varnothing.$$

The conditions (i) and (ii) mean that $Q_* = Q\,(\mathrm{Pref}(T_i), E_i)$. There are two cases to consider.

*Case 1.* $T_i \not\subseteq L(G_{i-1})$. Then $J_i = \mathrm{Pref}(T_i)$ and $N_i = Q_*$. The condition (iii′) then means that

$$\begin{aligned} P_i &= P'(N_i, E_i) \\ &= \{\, \langle\!\langle u \rangle\!\rangle \to a\, \langle\!\langle v \rangle\!\rangle \mid u, v \in Q_*, a \in \Sigma, u \backslash L_* \supseteq a\,(v \backslash L_*)\,\} \cup \\ &\quad\ \{\, \langle\!\langle u \rangle\!\rangle \to \varepsilon \mid u \in Q_*, u \in L_*\,\}. \end{aligned}$$

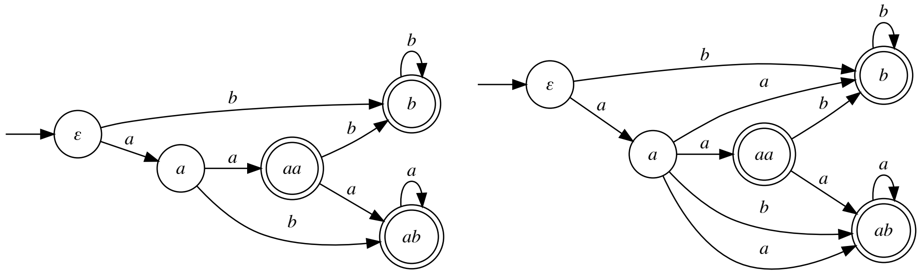

**Fig. 1.** The minimal DFA for $aba^* \cup bb^* \cup aa(a^* \cup b^*)$ (left) and its fattening (right).

The learner's hypothesis $G_i$ is just like $G_*$ (the right-linear grammar corresponding to the minimal DFA for $L_*$) except that it may have additional productions. In general, the finite automaton corresponding to $G_i$ is nondeterministic, but it accepts exactly the same strings as the minimal DFA. It is in fact the result of adding to the minimal DFA as many transitions as possible without changing the accepted language. (Let us call this NFA the *fattening* of the minimal DFA.) We have $L(G_i) = L_*$, and at all stages $l \geq i$, the sets $N_l$ and $P_l$, as well as the output grammar $G_l$, will stay constant.

*Case 2.* $T_i \subseteq L(G_{i-1})$. In this case, $J_i$ may be a proper subset of $\mathrm{Pref}(T_i)$. The condition (ii) implies that $N_i$ is in one-to-one correspondence with some subset of $Q_*$. That is to say, for each $u' \in N_i$, there is a $u \in Q_*$ such that $u' \backslash L_* = u \backslash L_*$, and if $u', v' \in N_i$ and $u' \backslash L_* = v' \backslash L_*$, then $u' = v'$. The condition (iii$'$) implies

$$\begin{aligned} P'(N_i, E_i) = \{\, \langle\!\langle u \rangle\!\rangle \to a\, \langle\!\langle v \rangle\!\rangle \mid u, v \in N_i, a \in \Sigma, u \backslash L_* \supseteq a\,(v \backslash L_*) \,\} \cup \\ \{\, \langle\!\langle u \rangle\!\rangle \to \varepsilon \mid u \in N_i, u \in L_* \,\}. \end{aligned}$$

This means that the NFA corresponding to this right-linear grammar is isomorphic to a subautomaton of the fattening of the minimal DFA, so we must have $L(G_i) \subseteq L_*$. If $L(G_i) = L_*$, then the learner's hypothesis will remain the same at all later stages. If $L_* - L(G_i) \neq \varnothing$, then Case 1 applies at the earliest stage $l \geq i$ such that $t_l \notin L(G_i)$.

*Example 1.* Suppose that the target language is $L_* = aba^* \cup bb^* \cup aa(a^* \cup b^*)$. The minimal DFA for $L_*$ and its fattening are shown in Figure 1. On receiving $\{b, aa, ab\}$ (presented in this order), Algorithm 2 outputs the right-linear grammar corresponding to the NFA on the right. On receiving $\{b, aba\}$, it outputs the right-linear grammar corresponding to the NFA in Figure 2. In both cases, the learner's hypothesis stays constant at all later stages.

Although Algorithm 2 is not set-driven and the grammar it stabilizes on depends on the order of the positive presentation, its behavior is not unreasonable. In a way, it tries to postulate as few nonterminals (states) as possible. It processes all positive examples immediately and does not engage in any "delaying trick" [3,4] just to achieve polynomial update time.

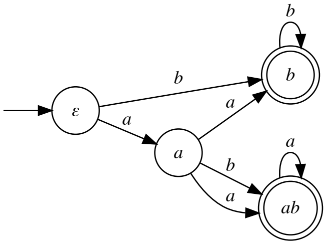

**Fig. 2.** An NFA for $aba^* \cup bb^* \cup aa(a^* \cup b^*)$.

Instead of (a) and (b), Algorithm 2 satisfies the following weaker conditions:

(a′) If $\{t_1, \dots, t_i\} \subseteq L(G_i)$ and $t_{i+1} \in \{t_1, \dots, t_i\}$, then $G_{i+1} = G_i$.
(b′) If $\{t_1, \dots, t_i\} \not\subseteq L(G_{i-1})$, then $\{t_1, \dots, t_i\} \subseteq L(G_i)$.

It is easy to verify (a′). For (b′), if $t_k = a_1 \dots a_n$ with $a_j \in \Sigma$ $(1 \leq j \leq n)$, then $\mathrm{Pref}(\{t_k\}) \subseteq J_i$ means that for each $j = 0, \dots, n$, there is a $u_j \in N_i$ such that $(\{\varepsilon\} \cup \Sigma)E_i \cap (u_j \backslash L_*) = (\{\varepsilon\} \cup \Sigma)E_i \cap (a_1 \dots a_j \backslash L_*)$. We have

$$\begin{aligned} a_{j+1}(E_i \cap (v_{j+1} \backslash L_*)) &= a_{j+1}(E_i \cap (a_1 \dots a_{j+1} \backslash L_*)) \\ &= a_{j+1}E_i \cap (a_1 \dots a_j \backslash L_*) \\ &= a_{j+1}E_i \cap (v_j \backslash L_*) \\ &\subseteq v_j \backslash L_*, \end{aligned}$$

so $P_i$ contains the production $\langle\!\langle v_j \rangle\!\rangle \to a_{j+1}\langle\!\langle v_{j+1} \rangle\!\rangle$. Since

$$\varepsilon \in \{\varepsilon\}E_i \cap (a_1 \dots a_n \backslash L_*) = \{\varepsilon\}E_i \cap (v_n \backslash L_*),$$

we have $v_n \in L_*$, which implies that $\langle\!\langle v_n \rangle\!\rangle \to \varepsilon$ is in $P_i$ as well. So we have a derivation $\langle\!\langle \varepsilon \rangle\!\rangle = \langle\!\langle v_0 \rangle\!\rangle \Rightarrow^*_{G_i} a_1 \dots a_n = t_k$.

We cannot prove that Algorithm 2 satisfies (d) (or (d†), for that matter) in the same way we proved (d) for Algorithm 1. This is because when $D \subseteq T_i$, where $D$ is a polynomial-sized set satisfying (i), (ii), and (iii′), we may have $T_i \subseteq L(G_{i-1})$, in which case the algorithm may need an additional string from $L_* - L(G_i)$ in order to reach a correct grammar. This additional string depends on the set $N_i \subset Q_*$, of which there are exponentially many possibilities.[6] We can summarize this behavior of Algorithm 2 as follows:

(d′) There is a finite set $D \subseteq L_*$ whose total size is bounded by a polynomial in the representation size of $G_*$ such that whenever $D \subseteq \{t_1, \dots, t_i\}$, there is a string $t$, depending on $(t_1, \dots, t_i)$, of length less than $|Q_*|$ such that whenever $t \in \{t_{i+1}, \dots, t_l\}$, $G_l$ is constant and isomorphic to the right-linear grammar corresponding to a subautomaton of the fattening of the minimal DFA corresponding to $G_*$.

[6] We only need to worry about *maximal* subsets of $Q_*$ such that the corresponding subautomaton of the fattening of the minimal DFA for $L_*$ fails to accept all strings in $L_*$. Still, there may be exponentially many such maximal sets.

Our learning algorithms for context-free languages resemble Algorithm 2 in many ways, but also differ in some important respects.

## 3 Context-Free Languages

Like Algorithms 1 and 2, our algorithms for learning context-free languages use membership queries to test whether a given string $x$ belongs to the string set associated with a postulated nonterminal. For this to be possible, we must assume that the set *reduces* in polynomial time to the target language $L_*$. The reduction must be uniform across different target languages—the learner must have a representation of a nonterminal without full knowledge of the target language, and this representation must determine the reduction by which the string set of the nonterminal reduces to the target language.

Let us formally define the three subclasses of context-free grammars we are interested in. If $G = (N, \Sigma, P, S)$ is a context-free grammar, a tuple $(X_B)_{B \in N}$ of sets in $\mathscr{P}(\Sigma^*)$ is a *pre-fixed point* of $G$ if for each production $A \to w_0\, B_1\, w_1\, \ldots\, B_n\, w_n$ in $P$, we have

$$X_A \supseteq w_0\, X_{B_1}\, w_1\, \ldots\, X_{B_n}\, w_n.$$

The tuple $(X_B)_{B \in N}$ with $X_B = \Sigma^*$ for all $B \in N$ is the greatest pre-fixed point of $G$, and the tuple $(L_G(B))_{B \in N}$ is the least pre-fixed point (under the partial order of componentwise inclusion). A pre-fixed point $(X_B)_{B \in N}$ is *sound* if $X_S \subseteq L(G)$ (or, equivalently, if $X_S = L(G)$). Let $\Gamma$ be a set of operations on $\mathscr{P}(\Sigma^*)$ (of varying arity). Then $G$ has the *$\Gamma$-closure property* if $G$ has a sound pre-fixed point (SPP) each of whose components belongs to the $\Gamma$-closure of $\mathrm{Quot}(L(G))$. Setting $\Gamma$ to $\{\cap\}$, we get the class of context-free grammars with the *intersection closure property.* With $\Gamma = \{\cap, \bar{\cdot}, \cup\}$ (intersection, complement, and union), we get the context-free grammars with the *Boolean closure property.* If we add to this set $\varnothing$, $\varepsilon$, and $a$ ($a \in \Sigma$) (considered the zero-ary operations producing $\varnothing$, $\{\varepsilon\}$, and $\{a\}$, respectively) and the concatenation and Kleene star operations, we get the context-free grammars with the *extended regular closure property.*

Our learning algorithms targeting context-free grammars with the $\Gamma$-closure property use expressions built from *query atoms* $(u, v)^{\triangleleft}$, where $u, v \in \Sigma^*$, and symbols for operations in $\Gamma$. These expressions denote subsets of $\Sigma^*$ relative to $L_*$ in the obvious way:

$$\begin{aligned}
\llbracket (u,v)^{\triangleleft} \rrbracket^{L_*} &= u \backslash L_* / v, & \llbracket \varnothing \rrbracket^{L_*} &= \varnothing, \\
\llbracket e_1 \cap e_2 \rrbracket^{L_*} &= \llbracket e_1 \rrbracket^{L_*} \cap \llbracket e_2 \rrbracket^{L_*}, & \llbracket \varepsilon \rrbracket^{L_*} &= \{\varepsilon\}, \\
\llbracket \overline{e_1} \rrbracket^{L_*} &= \Sigma^* - \llbracket e_1 \rrbracket^{L_*}, & \llbracket a \rrbracket^{L_*} &= \{a\}, \\
\llbracket e_1 \cup e_2 \rrbracket^{L_*} &= \llbracket e_1 \rrbracket^{L_*} \cup \llbracket e_2 \rrbracket^{L_*}, & \llbracket e_1 e_2 \rrbracket^{L_*} &= \llbracket e_1 \rrbracket^{L_*} \llbracket e_2 \rrbracket^{L_*}, \\
& & \llbracket e_1^* \rrbracket^{L_*} &= (\llbracket e_1 \rrbracket^{L_*})^*.
\end{aligned}$$

An important property of these expressions is the following. If $e$ is an expression, let $C_e = \{\, (u, v) \mid (u, v)^{\triangleleft} \text{ occurs in } e \,\}$.

($*$) The truth value of $x \in \llbracket e \rrbracket^{L_*}$ only depends on the truth values of $y \in u \backslash L_* / v$ for substrings $y$ of $x$ and $(u, v) \in C_e$.

### 3.1 Examples

Let us look at some examples.[7] If $A$ is a nonterminal of a grammar $G$, we often abuse the notation and write just $A$ for the set $L_G(A)$.

*Example 2.* Consider

$$L_1 = \{\, a^m b^n \mid m \text{ is even and } m = n \,\} \cup \{\, a^m b^n \mid m \text{ is odd and } 2m = n \,\}.$$

This language is generated by the following grammar:

$$S \to T \mid U, \quad T \to \varepsilon \mid aaTbb, \quad U \to abb \mid aaUbbbb.$$

We have

$$\begin{aligned} S &= L_1 = \llbracket (\varepsilon, \varepsilon)^{\triangleleft} \rrbracket^{L_1}, \\ T &= \{\, a^m b^n \mid m \text{ is even and } m = n \,\} = \llbracket (\varepsilon, \varepsilon)^{\triangleleft} \cap (aa, bb)^{\triangleleft} \rrbracket^{L_1}, \\ U &= \{\, a^m b^n \mid m \text{ is odd and } 2m = n \,\} = \llbracket (\varepsilon, \varepsilon)^{\triangleleft} \cap (aa, bbbb)^{\triangleleft} \rrbracket^{L_1}. \end{aligned}$$

So this grammar has the intersection closure property.

*Example 3.* Consider

$$L_2 = \{\, a^m b^n \mid m \text{ is even and } m = n \,\} \cup \{\, a^m b^n \mid m \text{ is odd and } 2m \le n \,\}.$$

This language is generated by the following grammar:

$$S \to T \mid UB, \quad T \to \varepsilon \mid aaTbb, \quad U \to abb \mid aaUbbbb, \quad B \to \varepsilon \mid Bb.$$

We have

$$\begin{aligned} S &= L_2 = \llbracket (\varepsilon, \varepsilon)^{\triangleleft} \rrbracket^{L_2}, \\ T &= \{\, a^m b^n \mid m \text{ is even and } m = n \,\} = \left\llbracket (\varepsilon, \varepsilon)^{\triangleleft} \cap \overline{(aa, bbbb)^{\triangleleft}} \right\rrbracket^{L_2}, \\ U &= \{\, a^m b^n \mid m \text{ is odd and } 2m = n \,\} = \left\llbracket (\varepsilon, \varepsilon)^{\triangleleft} \cap (aa, bbbb)^{\triangleleft} \cap \overline{(aa, bbb)^{\triangleleft}} \right\rrbracket^{L_2}, \\ B &= b^* = \llbracket (abb, \varepsilon) \rrbracket^{L_2}. \end{aligned}$$

So this grammar has the Boolean closure property. We can show that the set $T$ is not in the intersection closure of $\mathrm{Quot}(L_2)$, and indeed, $L_2$ has no grammar with the intersection closure property.

[7] In all of these examples, the SPP witnessing the relevant closure property is the least SPP, but there are cases where the witnessing SPP cannot be the least one [5].

*Example 4.* Consider

$$L_3 = \{\, a^m b^n \mid m \text{ is even and } m \geq n \,\} \cup \{\, a^m b^n \mid m \text{ is odd and } 2m \leq n \leq 3m \,\}.$$

This language is generated by the following grammar:

$$\begin{gathered} S \to AT \mid U, \quad A \to \varepsilon \mid Aaa, \quad T \to \varepsilon \mid aaTbb, \\ U \to abb \mid V \mid aaUbbbb, \quad V \to abbb \mid aaVbbbbbb. \end{gathered}$$

We have

$$\begin{aligned} &S = L_3 = \llbracket (\varepsilon,\varepsilon)^{\triangleleft} \rrbracket^{L_3}, \\ &A = (aa)^* = \llbracket (\varepsilon, aabb)^{\triangleleft} \rrbracket^{L_3} \\ &T = \{\, a^m b^n \mid m \text{ is even and } m = n \,\} = \left\llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap (aa)^* b^* \cap \overline{(\varepsilon, b)^{\triangleleft}} \right\rrbracket^{L_3}, \\ &U = \{\, a^m b^n \mid m \text{ is odd and } 2m \leq n \leq 3m \,\} = \llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap (aa)^* ab^* \rrbracket^{L_3}, \\ &V = \{\, a^m b^n \mid m \text{ is odd and } n = 3m \,\} = \left\llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap (aa)^* ab^* \cap \overline{(\varepsilon, b)^{\triangleleft}} \right\rrbracket^{L_3} \end{aligned}$$

So this grammar has the extended regular closure property. We can show that $L_3$ has no grammar with the Boolean closure property.

*Example 5.* Consider

$$L_4 = \{\, a^m b^n \mid m \geq n \,\} \cup \{\, a^m b^n \mid m \geq 1 \text{ and } 2m \leq n \leq 3m \,\}.$$

Unlike the previous three examples, this is not a deterministic context-free language. It is generated by the following unambiguous grammar:

$$\begin{gathered} S \to AT \mid U, \quad A \to \varepsilon \mid Aa, \quad T \to \varepsilon \mid aTb, \\ U \to abb \mid V \mid aUbb, \quad V \to abbb \mid aVbbb. \end{gathered}$$

We have

$$\begin{aligned} &S = L_4 = \llbracket (\varepsilon,\varepsilon)^{\triangleleft} \rrbracket^{L_4}, \\ &A = a^* = \llbracket (\varepsilon, ab)^{\triangleleft} \rrbracket^{L_4}, \\ &T = \{\, a^m b^n \mid m = n \,\} = \left\llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap \left( \varepsilon \cup \overline{(a, bb)^{\triangleleft}} \right) \right\rrbracket^{L_4}, \\ &U = \{\, a^m b^n \mid m \geq 1 \text{ and } 2m \leq n \leq 3m \,\} = \left\llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap \left( ab \cup \overline{(\varepsilon, \varepsilon)^{\triangleleft}} \right) bb^* \right\rrbracket^{L_4}, \\ &V = \{\, a^m b^n \mid n = 3m \,\} = \left\llbracket (\varepsilon,\varepsilon)^{\triangleleft} \cap \left( ab \cup \overline{(\varepsilon, \varepsilon)^{\triangleleft}} \right) bb^* \cap \overline{(\varepsilon, b)^{\triangleleft}} \right\rrbracket^{L_4}. \end{aligned}$$

So this grammar has the extended regular closure property. Again, we can show that $L_4$ has no grammar with the Boolean closure property.

*Example 6.* The inherently ambiguous language

$$\{\, a^l b^m c^n d^q \mid l, m, n, q \geq 1 \text{ and } l = n \vee m > q \,\}$$

does not have a grammar with the extended regular closure property [7].

### 3.2 Algorithm

Let us describe our algorithm for learning context-free languages, leaving the set $\Gamma$ of available operations as an unspecified parameter. The set $\Gamma$ may be any subset of the Boolean and regular operations, as long as it contains intersection. In this algorithm, quotients of $L_*$ will play a role similar to the role left quotients played in Algorithm 2. An important difference is that since a context-free language has infinitely many quotients (unless it is regular), the learner cannot identify the set of all quotients, even in the limit. It tries to identify a superset of the set of quotients the target grammar "uses", so to speak, and this is done by a strategy similar to Algorithm 2.

Since the learner can only postulate a polynomial number of nonterminals, we place an arbitrary finite bound $k$ on the number of occurrences of symbols for operations in the representation of a nonterminal. Since polynomially many string pairs $(u, v)$ are available as building blocks of nonterminals, this is a necessary restriction.

We also have to place a suitable syntactic restriction on nonterminals that ensures that their denotations relative to $L_*$ are included in $\mathrm{Sub}(L_*) = \{\, x \mid \text{for some } (u, v),\ uxv \in L_* \,\}$. We call a nonterminal obeying this restriction *guarded*:

- $(u, v)^{\triangleleft}$ is guarded.
- If $e_1$ is guarded, so is $e_1 \cap e_2$.
- If $e_1$ and $e_2$ are guarded, so is $e_1 \cup e_2$.

If $C$ is a finite subset of $\Sigma^* \times \Sigma^*$, we write $\mathcal{E}_k(C, \Gamma)$ for the set of all guarded expressions built up from query atoms in $\{\, (u, v)^{\triangleleft} \mid (u, v) \in C \,\}$ using operations in $\Gamma$ up to $k$ times.[8]

As for productions, we place a bound $r$ on the number of occurrences of nonterminals and a bound $s$ on the length of contiguous terminal strings on the right-hand side of a production. So a production postulated by the learner is of the form

$$A \to w_0\, B_1\, w_1\, \ldots\, B_n\, w_n,$$

where $n \leq r$ and $|w_i| \leq s$ $(0 \leq i \leq n)$.[9] The notation $\langle \Sigma^{\leq s} \rangle^{\leq r+1}$ is used to denote the set of possible choices of $(w_0, w_1, \ldots, w_n)$. Note that if $E \subseteq \Sigma^*$ is closed under substring, then the set

$$\Sigma^{\leq s}(E\, \Sigma^{\leq s})^{\leq r} = \bigcup \{\, w_0\, E\, w_1\, \ldots\, E\, w_n \mid (w_0, w_1, \ldots, w_n) \in \langle \Sigma^{\leq s} \rangle^{\leq r+1} \,\}$$

is also closed under substring, and is a superset of $E$.

[8] A reasonable optimization is to put expressions in some suitable "normal form", to avoid including a large number of equivalent expressions in $\mathcal{E}_k(C, \Gamma)$.

[9] An alternative is to allow arbitrary terminal strings to surround nonterminals on the right-hand side of productions, as long as they are "observed" in the positive data [5,6,7].

Define a strict total order $\prec_2$ on $\Sigma^* \times \Sigma^*$ by

$$(u_1, u_2) \prec_2 (v_1, v_2) \Longleftrightarrow u_1 \prec v_1 \vee (u_1 = v_1 \wedge u_2 \prec v_2).$$

We write $\preceq_2$ for the reflexive counterpart of $\prec_2$. For $J \subseteq \Sigma^* \times \Sigma^*$ and $E \subseteq \Sigma^*$, let

$$\begin{aligned} &Q^{r,s}(J, E) = \\ &\quad \{ (u_1, u_2) \mid (u_1, u_2) \in J \text{ and for each } (v_1, v_2) \in J, \\ &\qquad\qquad \text{if } (v_1, v_2) \prec_2 (u_1, u_2), \text{ then} \\ &\qquad\qquad (\Sigma^{\leq s}(E\,\Sigma^{\leq s})^{\leq r}) \cap (v_1 \backslash L_* / v_2) \neq (\Sigma^{\leq s}(E\,\Sigma^{\leq s})^{\leq r}) \cap (u_1 \backslash L_* / u_2) \}. \end{aligned}$$

For $K, E \subseteq \Sigma^*$ and a set $N$ of expressions, define

$$\begin{aligned} \mathrm{Sub}(K) &= \{ y \in \Sigma^* \mid uyv \in K \text{ for some } (u, v) \in \Sigma^* \times \Sigma^* \}, \\ \mathrm{Con}(K) &= \{ (u, v) \in \Sigma^* \times \Sigma^* \mid uyv \in K \text{ for some } y \in \Sigma^* \}, \\ P^{r,s}(N, E) &= \{ A \to w_0\, B_1\, w_1\, \ldots\, B_n\, w_n \mid \\ &\qquad 0 \leq n \leq r, A, B_1, \ldots, B_n \in N, \\ &\qquad (w_0, w_1, \ldots, w_n) \in \langle \Sigma^{\leq s} \rangle^{\leq r+1}, \\ &\qquad [\![A]\!]^{L_*} \supseteq w_0\, (E \cap [\![B_1]\!]^{L_*})\, w_1\, \ldots\, (E \cap [\![B_n]\!]^{L_*})\, w_n \}. \end{aligned}$$

The inclusion $[\![A]\!]^{L_*} \supseteq w_0\, (E \cap [\![B_1]\!]^{L_*})\, w_1\, \ldots\, (E \cap [\![B_n]\!]^{L_*})\, w_n$ in the definition of $P^{r,s}(N, E)$ is analogous to the inclusion $u \backslash L_* \supseteq a\,(E \cap (v \backslash L_*))$ in the definition of $P'(J, E)$ in (2).

With the necessary definitions in place, we can list the learning algorithm in Algorithm 3.

By the property $(*)$ of expressions used by the learner, membership queries that are needed to compute $N_i$ and $P_i$ are all of the form "$uyv \in L_*$?", where $(u, v) \in \mathrm{Con}(T_i)$ and $y \in \Sigma^{\leq s}(\mathrm{Sub}(T_i)\Sigma^{\leq s})^{\leq r}$. There are only polynomially many of them (in the total lengths of the strings in $T_i$).

Algorithm 3 is by no means capable of learning all context-free languages. What is the class of context-free languages that the algorithm can learn? Recall that the denotation of each nonterminal $B$ (relative to $L_*$) is a subset of $\mathrm{Sub}(L_*)$. Because of how $P^{r,s}(N, E)$ is defined, if Algorithm 3 stabilizes on a grammar $G_i$, then all its productions $A \to w_0\, B_1\, w_1\, \ldots\, B_n\, w_n$ are *valid* in the sense that

$$[\![A]\!]^{L_*} \supseteq w_0\ [\![B_1]\!]^{L_*}\ w_1\ \ldots\ [\![B_n]\!]^{L_*}\ w_n\,, \tag{3}$$

since $\bigcup_i E_i = \mathrm{Sub}(L_*)$. This means that the tuple of sets $([\![B]\!]^{L_*})_{B \in N_i}$ is a pre-fixed point of $G_i$. So $L_{G_i}(B) \subseteq [\![B]\!]^{L_*}$. In particular, $L(G_i) = L_{G_i}((\varepsilon, \varepsilon)^{\triangleleft}) \subseteq [\![(\varepsilon, \varepsilon)^{\triangleleft}]\!]^{L_*} = L_*$. The converse inclusion can be shown by appealing to the fact that Algorithm 3 satisfies the property (b′), which can be proved in the same way as with Algorithm 2. If $L_* \not\subseteq L(G_i)$, then at the earliest stage $l > i$ such that $t_l \notin L(G_i)$, we must have $G_l \neq G_i$, since $t_l \in L(G_l)$ by (b′). This contradicts the assumption that Algorithm 3 stabilizes on $G_i$. So we have $L(G_i) = L_*$ and $([\![B]\!]^{L_*})_{B \in N_i}$ is an SPP of $G_i$. This shows that $G_i$ has the $\Gamma$-closure property.

**Algorithm 3:** Learner for a subclass of the context-free languages.

```
Parameters: Positive integers $r, s, k$
Data: A positive presentation $t_1, t_2, \dots$ of $L_* \subseteq \Sigma^*$; membership oracle for
      $L_*$;
Result: A sequence of grammars $G_1, G_2, \dots$;
$T_0 := \varnothing$; $E_0 := \varnothing$; $J_0 := \varnothing$; $G_0 := (\{(\varepsilon, \varepsilon)^{\lhd}\}, \Sigma, \varnothing, (\varepsilon, \varepsilon)^{\lhd})$;
for $i = 1, 2, \dots$ do
    $T_i := T_{i-1} \cup \{t_i\}$; $E_i := E_{i-1} \cup \mathrm{Sub}(\{t_i\})$;
    if $T_i \subseteq L(G_{i-1})$ then
        $J_i := J_{i-1}$;
    else
        $J_i := \mathrm{Con}(T_i)$;
    end
    $N_i := \mathcal{E}_k(Q^{r,s}(J_i, E_i), \Gamma)$;
    $P_i := P^{r,s}(N_i, E_i)$;
    output $G_i := (N_i, \Sigma, P_i, (\varepsilon, \varepsilon)^{\lhd})$;
end
```

**Theorem 7.** *If, given a positive presentation of $L_*$ and the membership oracle for $L_*$, Algorithm 3 stabilizes on a grammar $G$, then $L(G) = L_*$ and $G$ has the $\Gamma$-closure property.*

We have seen that Algorithm 3 can only learn a context-free language that has a grammar with the $\Gamma$-closure property. Conversely, if $L_*$ has a grammar $G_*$ with the $\Gamma$-closure property, let $r$ and $s$ be the maximal number of nonterminals and the maximal length of contiguous terminal strings, respectively, on the right-hand side of productions of $G_*$. Let $k$ be the least number such that $G_*$ has an SPP each of whose components is denoted by a guarded expression containing at most $k$ operations in $\Gamma$. Then with this choice of $r, s, k$, one can show that Algorithm 3 learns $L_*$. The proof is similar to the proof for Algorithm 2, with some additional complications.[10]

Let $Q_*$ be the set of pairs $(u, v)$ of strings such that the query atom $(u, v)^{\lhd}$ occurs in expressions for components of the SPP for $G_*$. So if $N_*$ is the set of nonterminals of $G_*$, there is an expression $e_B \in \mathcal{E}_k(Q_*, \Gamma)$ for each $B \in N_*$ such that $(\llbracket e_B \rrbracket^{L_*})_{B \in N_*}$ is an SPP of $G_*$. We can safely assume that if $(u, v) \in Q_*$ and $(u', v') \prec_2 (u, v)$, then $u' \backslash L_* / v' \neq u \backslash L_* / v$. In particular, if $(u_1, v_1)$ and $(u_2, v_2)$ are distinct elements of $Q_*$, we have $u_1 \backslash L_* / v_1 \neq u_2 \backslash L_* / v_2$.

Now assume that $t_1, t_2, \dots$ is a positive presentation of $L_*$, and suppose $T_i = \{t_1, \dots, t_i\}$ is such that $Q_* \subseteq \mathrm{Con}(T_i)$.

*Case 1.* $T_i \not\subseteq L(G_{i-1})$. Then $Q_* \subseteq J_i = \mathrm{Con}(T_i)$. At each stage $l \geq i$, we have $Q_* \subseteq J_l$, and for each $(u, v) \in Q_*$, there is a $(u', v') \in Q^{r,s}(J_l, E_l)$ such

[10] Unlike the algorithms in [9,5,6,7], Algorithm 3 tries to avoid using different query atoms that denote the same quotient of $L_*$. This should be compared with Leiß's [8] algorithm for the case $\Gamma = \{\cap\}$, which tries to minimize the number of nonterminals used.

that $(u',v') \preceq_2 (u,v)$ and

$$(\varSigma^{\le s}(E_l\varSigma^{\le s})^{\le r}) \cap (u'\backslash L_*/v') = (\varSigma^{\le s}(E_l\varSigma^{\le s})^{\le r}) \cap (u\backslash L_*/v). \tag{4}$$

For each $B \in N_*$, let $e'_B$ be the result of replacing each query atom $(u,v)^{\triangleleft}$ in $e_B$ by $(u',v')^{\triangleleft}$. Then $e'_B \in N_l$ for each $B \in N_*$, and (4) implies

$$(\varSigma^{\le s}(E_l\varSigma^{\le s})^{\le r}) \cap [\![e_{B'}]\!]^{L_*} = (\varSigma^{\le s}(E_l\varSigma^{\le s})^{\le r}) \cap [\![e_B]\!]^{L_*}\,. \tag{5}$$

For each production

$$A \to w_0\, B_1\, w_1\, \ldots\, B_n\, w_n$$

of $G_*$, we have

$$[\![e_A]\!]^{L_*} \supseteq w_0\ [\![e_{B_1}]\!]^{L_*}\ w_1\ \ldots\ [\![e_{B_n}]\!]^{L_*}\ w_n,$$

since $([\![e_B]\!]^{L_*})_{B\in N_*}$ is an SPP of $G_*$. Since $(w_0,w_1,\ldots,w_n) \in \langle \varSigma^{\le s}\rangle^{\le r+1}$, we have

$$(\varSigma^{\le s})(E_l\varSigma^{\le s})^{\le r} \cap [\![e_A]\!]^{L_*} \supseteq w_0\,(E_l \cap [\![e_{B_1}]\!]^{L_*})\,w_1\,\ldots\,(E_l \cap [\![e_{B_n}]\!]^{L_*})\,w_n.$$

But since $E_l \subseteq \varSigma^{\le s}(E_l\varSigma^{\le s})^{\le r}$, this together with (5) implies

$$(\varSigma^{\le s})(E_l\varSigma^{\le s})^{\le r} \cap [\![e_{A'}]\!]^{L_*} \supseteq w_0\,(E_l \cap [\![e_{B'_1}]\!]^{L_*})\,w_1\,\ldots\,(E_l \cap [\![e_{B'_n}]\!]^{L_*})\,w_n,$$

so

$$e_{A'} \to w_0\, e_{B'_1}\, w_1\, \ldots\, e_{B'_n}\, w_n$$

is a production in $P_l$. Since $(\varepsilon,\varepsilon)' = (\varepsilon,\varepsilon)$ and $((\varepsilon,\varepsilon)^{\triangleleft})' = (\varepsilon,\varepsilon)^{\triangleleft}$, this means that $G_l$ contains a homomorphic image of $G_*$, and so $L_* \subseteq L(G_l)$. It follows that for all $l \ge i$, we have $J_l = J_i$ and $Q^{r,s}(J_l,E_l) \subseteq J_i$. Since $Q^{r,s}(J,E)$ is monotone in $E$, we see that $Q^{r,s}(J_l,E_l)$ eventually stabilizes, i.e., there is an $m \ge i$ such that for all $l \ge m$, $Q^{r,s}(J_l,E_l) = Q^{r,s}(J_m,E_m)$. Then we have $N_l = N_m$ and $P_l = P^{r,s}(N_m,E_l)$ for all $l \ge m$. Since $P^{r,s}(N,E)$ is antitone in $E$, it follows that $P_l$, and hence $G_l$, eventually stabilize. By Theorem 7, the grammar that the algorithm stabilizes on is a grammar for $L_*$.

*Case 2.* $T_i \subseteq L(G_{i-1})$. We distinguish two cases.

*Case 2.1.* $T_l \subseteq L(G_{l-1})$ for all $l \ge i$. Then $J_l = J_i = J_{i-1}$ for all $l \ge i$, and by a reasoning similar to Case 1, $G_l$ will eventually stabilize to a correct grammar for $L_*$.

*Case 2.2.* $T_l \not\subseteq L(G_{l-1})$ for some $l \ge i$. Then we are in Case 1 at the earliest stage $l \ge i$ when this happens.

**Theorem 8.** *Suppose that $G_* = (N_*,\varSigma,P_*,S)$ is a context-free grammar with the $\varGamma$-closure property such that if $B \to w_0\, B_1\, w_1\, \ldots,\, B_n\, w_n$ is a production in $P_*$, then $n \le r$ and $|w_j| \le s$ $(0 \le j \le n)$. Let $L_* = L(G_*)$. Given a positive presentation of $L_*$ and the membership oracle for $L_*$, Algorithm 3 converges to a grammar for $L_*$ with the $\varGamma$-closure property.*

Let us close by stating a data efficiency property of Algorithm 3 in the style of (d′). Let $D$ be the subset of $L_*$ with the least cardinality such that $Q_* \subseteq \mathrm{Con}(D)$. Then $|D| \leq |Q_*| \leq k|N_*|$, but there may be strings in $D$ whose length is exponential in the representation size of $G_*$. Suppose $D \subseteq T_i = \{t_1, \dots, t_i\}$.

*Case 1.* If $T_i \nsubseteq L(G_{i-1})$, then the size of $J_i = \mathrm{Con}(T_i)$ is polynomial in the total lengths of the strings in $T_i$. We need additional positive examples to make the set of nonterminals postulated by the learner equal $\mathcal{E}_k(Q^{r,s}(J_i, \mathrm{Sub}(L_*)), \mathit{\Gamma})$. We need at most $\binom{|J_i|}{2}$ such positive examples. We also need positive examples to eliminate any productions that are not valid in the sense of (3). The number of such productions is at most polynomial in $|J_i|$. We can combine these two types of positive examples and present them in any order, interspersed with other positive examples, to force the learner to stabilize on a correct grammar.

*Case 2.* $T_i \subseteq L(G_{i-1})$. Then $J_i$ may be a proper subset of $\mathrm{Con}(T_i)$. As soon as we have $T_l \nsubseteq L(G_{l-1})$ $(i \leq l)$, we will be in Case 1. Until this happens, polynomially many positive examples (in the size of $J_i$) suffice to make the set of nonterminals postulated by the learner equal $\mathcal{E}_k(Q^{r,s}(J_i, \mathrm{Sub}(L_*)), \mathit{\Gamma})$ and to eliminate all invalid productions formed with these nonterminals. When $T_l$ $(l \geq i)$ contains these positive examples and we are still in Case 2, we have two possibilities.

*Case 2.1.* $L_* \subseteq L(G_l)$. In this case, $G_l$ has already stabilized.

*Case 2.2.* $L_* \nsubseteq L(G_l)$. Then any positive example not in $L(G_l)$ puts us in Case 1.

Summarizing, all we seem to be able to say about Algorithm 3 is that it satisfies the following complex property:

(d†′) There exists a finite subset $D$ of $L_*$ whose cardinality is bounded by a polynomial in the representation size of $G_*$ such that

- – whenever $D \subseteq \{t_1, \dots, t_i\} \subseteq L_*$, one can find a finite subset $D'$ of $L_*$, depending on $(t_1, \dots, t_i)$, whose cardinality is polynomial in the total lengths of the strings in $\{t_1, \dots, t_i\}$ such that
  - • whenever $D' \subseteq \{t_{i+1}, \dots, t_k\} \subseteq L_*$, one can find a string $t \in L_*$, depending on $(t_1, \dots, t_k)$, such that
    - ∗ whenever $t \in \{t_{k+1}, \dots, t_l\} \subseteq L_*$, one can find a finite subset $D''$ of $L_*$, depending on $(t_1, \dots, t_l)$, whose cardinality is polynomial in the total lengths of the strings in $\{t_1, \dots, t_l\}$ such that
      - · whenever $D'' \subseteq \{t_{l+1}, \dots, t_m\} \subseteq L_*$, the output $G_m$ of Algorithm 3 is constant and is a correct grammar for $L_*$.

**Acknowledgements** I am grateful to Ryo Yoshinaka for the joint work this paper is an outgrowth of. This work was supported by JSPS KAKENHI Grant Number 17K00026.

## References

1. Angluin, D.: A note on the number of queries needed to identify regular languages. Information and Control **51**(1), 76–87 (1981). https://doi.org/10.1016/S0019-9958(81)90090-5
2. Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation **75**(2), 87–106 (1987). https://doi.org/10.1016/0890-5401(87)90052-6
3. Case, J.: Gold-style learning theory. In: Heinz, J., Sempere, J.M. (eds.) Topics in Grammatical Inference, pp. 1–23. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-48395-4_1
4. Eyraud, R., Heinz, J., Yoshinaka, R.: Efficiency in the identification in the limit learning paradigm. In: Heinz, J., Sempere, J.M. (eds.) Topics in Grammatical Inference, pp. 25–46. Springer Berlin Heidelberg, Berlin, Heidelberg (2016). https://doi.org/10.1007/978-3-662-48395-4_2
5. Kanazawa, M., Yoshinaka, R.: The strong, weak, and very weak finite context and kernel properties. In: Drewes, F., Martín-Vide, C., Truthe, B. (eds.) Language and Automata Theory and Applications. pp. 77–88. Springer International Publishing, Cham (2017). https://doi.org/10.1007/978-3-319-53733-7_5
6. Kanazawa, M., Yoshinaka, R.: A hierarchy of context-free languages learnable from positive data and membership queries. In: Chandlee, J., Eyraud, R., Heinz, J., Jardine, A., van Zaanen, M. (eds.) Proceedings of the Fifteenth International Conference on Grammatical Inference. Proceedings of Machine Learning Research, vol. 153, pp. 18–31. PMLR (23–27 Aug 2021), https://proceedings.mlr.press/v153/kanazawa21a.html
7. Kanazawa, M., Yoshinaka, R.: Extending distributional learning from positive data and membership queries. In: Proceedings of the Sixteenth International Conference on Grammatical Inference (to appear)
8. Leiß, H.: Learning context free grammars with the finite context property: A correction of A. Clark's algorithm. In: Morrill, G., Muskens, R., Osswald, R., Richter, F. (eds.) Formal Grammar 2014. pp. 121–137. Lecture Notes in Computer Science, Springer, Berlin, Heidelberg (2014). https://doi.org/10.1007/978-3-662-44121-3_8
9. Yoshinaka, R.: Towards dual approaches for learning context-free grammars based on syntactic concept lattices. In: Mauri, G., Leporati, A. (eds.) Developments in Language Theory, 15th International Conference, DLT 2011. pp. 429–440. Lecture Notes in Computer Science, Springer, Berlin (2011). https://doi.org/10.1007/978-3-642-22321-1_37