<table>
<thead>
<tr>
<th>Title</th>
<th>Very Simple Grammars and Polynomial-Time Learning</th>
</tr>
</thead>
<tbody>
<tr>
<td>Author(s)</td>
<td>Yokomori, Takashi</td>
</tr>
<tr>
<td>Citation</td>
<td>数理解析研究所講究録 (1991), 754: 15-24</td>
</tr>
<tr>
<td>Issue Date</td>
<td>1991-06</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/82121">http://hdl.handle.net/2433/82121</a></td>
</tr>
<tr>
<td>Type</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>Textversion</td>
<td>Publisher</td>
</tr>
</tbody>
</table>

Kyoto University
Very Simple Grammars and Polynomial-Time Learning*

Takashi YOKOMORI

Department of Computer Science and Information Mathematics
University of Electro-Communications
1-5-1 Chofugaoka, Chofu, Tokyo 182, JAPAN
E-mail: yokomori%cs.uec.ac.jp@relay.cs.net

Abstract. This paper concerns a subclass of simple deterministic grammars, called very simple grammars, and studies the problem of identifying the subclass in the limit from positive data. The class of very simple languages forms a proper subclass of simple deterministic languages and is incomparable to the class of regular languages.

Besides some characterization results for very simple languages, we show that the class of very simple grammars is polynomial time identifiable in the limit from positive data in the sense of Pitt. That is, there is an algorithm that, given the targeted very simple grammar \( G_* \), identifies a very simple grammar \( G \) equivalent to \( G_* \) in the limit from positive data, satisfying the property that the time for updating a conjecture is bounded by a polynomial in \( m \), and the total number of prediction errors made by the algorithm is bounded by the cardinality \( k \) of the terminal alphabet involved in \( G_* \), where \( m \) is the maximum of \( k \) and the lengths of all positive data provided.

As corollaries, it immediately follows that the class of very simple grammars is identifiable in polynomial time via equivalence queries only and it is also PAC-identifiable in polynomial time.

1 Introduction

Since the class of regular languages has been shown to be efficiently identifiable using deterministic finite-state automata (DFAs) from what is called "minimally adequate teacher" by Angluin ([3]), a computational approach to learning theory or, more specifically, to grammatical inference has been again receiving much attention, and intensive works on this issue have been reported.

From the practical point of view, there are, we believe, two major requirements in the study of inductive inference algorithm for formal languages. That is, the identification algorithm must have a good time efficiency' and run only with positive data (examples).

Angluin [1] has given several conditions for the class of languages to be identifiable in the limit from positive data, and presented some examples of identifiable classes. She has also proposed sub-
classes of regular languages called $k$-reversible languages for each $k \geq 0$ and shown these classes are identifiable in the limit from positive data with the polynomial time of updating conjectures ([2]).

Motivated by a question posed by Angluin, however, one natural question has been quite recently recognized as significant: In what sense we should analyse the time complexity of an "in-the-limit" algorithm? Because one may define the notion of polynomial-time identification in the limit in various ways. And, it was not until quite recently that the polynomial-time identifiability in the limit was reasonably defined by Pitt. He proposed the following definition ([10]), which is one of the newest definitions ever proposed for the polynomial-time identifiability in the limit.

Informally, we say a class of languages $C$ is identifiable in the limit in polynomial time using a class of representations $\mathcal{R}$ iff there is an algorithm $A$ which, given $L$ in $C$, identifies $r$ in $\mathcal{R}$ representing $L$ in the limit, with the property that there exist polynomials $p$ and $q$ such that for any $n$, for any $L$ for which a correct representation is of size $n$, the number of times $A$ makes a wrong conjecture is at most $p(n)$, and the time for updating a conjecture is at most $q(n, N)$, where $N$ is the sum of lengths of data provided. In [15] it is shown that the subclass of regular languages called strictly $k$-testable languages is identifiable in the limit in polynomial time from positive data.

This paper deals with a class of grammars called very simple grammars and discusses the identification problem of the class of very simple grammars. To author's knowledge, the notion of a very simple grammar was originally introduced in [5] in the study on some type of Thue systems and the equivalence problem. The class of very simple languages forms a proper subclass of simple deterministic languages by Korenjak and Hopcroft([7]), and is incomparable to the class of regular languages.

After providing some of characterization results for very simple languages, we show that the class of very simple grammars is identifiable in the limit in polynomial time in the sense of Pitt. In fact, the identification of the class is achieved using only positive data.

The main result in this paper provides the first instance of language class containing non-regular languages which is identifiable in the limit in polynomial time in the sense of Pitt.

As corollaries, it immediately follows that the class of very simple grammars is identifiable in polynomial time via equivalence queries only and it is also PAC-identifiable in polynomial time.

2 Definitions

We assume the reader to be familiar with the rudiments of automata and formal language theory. (For notions and notations not stated here, see, e.g., [6].)

Let $\mathcal{R}$ be a class of representations for a class of languages $C$ to be identified. Given an $r$ in $\mathcal{R}$, $L(r)$ denotes the language represented by $r$.

For a given $r \in \mathcal{R}$, a presentation of the language $L(r)$ is any infinite sequence of data such that every $w \in \Sigma^*$ occurs at least once in the sequence with its sign (+: when $w \in L(r)$, or −: otherwise), and no other data (incorrectly labeled) appear in the sequence. A positive presentation of $L(r)$ is an infinite sequence of data such that every $w \in L(r)$ occurs at least once in the sequence and no other data not in $L(r)$ appear in the sequence.

Let $r$ be a representation in $\mathcal{R}$ representing a given $L$(i.e., $L = L(r)$). An algorithm $A$ is said to identify a language $L$ in the limit using $\mathcal{R}$ iff for any presentation of $L$, the infinite sequence of representations $r_i$ in $\mathcal{R}$ produced by $A$ satisfies the property that there exists a representation $r'$ in $\mathcal{R}$ such that for all sufficiently large $i$, the $i$-th conjecture (representation) $r_i$ is identical to $r'$ and $L(r') = L(r)$. A class of languages $C$ is identifiable in the limit using $\mathcal{R}$ iff there exists an algorithm $A$ that, given an $L$ in $C$, identifies $L$ in the limit using $\mathcal{R}$.

Let $A$ be an algorithm for identifying a language class $C$ in the limit using $\mathcal{R}$ and let $L$ be a language in $C$. Suppose that after examining $i$ data, the algorithm $A$ conjectures some representation $r_i$ for $L$. We say that $A$ makes an implicit error of prediction at step $i$ if $r_i$ is not consistent with the $(i+1)$-st example.

[Polynomial-time Identification in the limit][10]

A class $C$ is identifiable in the limit in polynomial time using $\mathcal{R}$ iff there exists an algorithm
\[ A \text{ for identifying } C \text{ in the limit using } \mathcal{R} \text{ with the property that there exist polynomials } p \text{ and } q \text{ such that for any } n, \text{ for any } L \text{ for which a correct representation is of size } n, \text{ and for any presentation of } L, \text{ the number of implicit errors of prediction made by } A \text{ is at most } p(n), \text{ and the time used by } A \text{ between receiving the } i\text{-th example } w_i \text{ and outputting the } i\text{-th conjectured representation } r_i \text{ is at most } q(n, m_1 + \ldots + m_i), \text{ where } m_j = \text{lg}(w_j). \]

Finally, we say that \( \mathcal{R} \) is identifiable in the limit in polynomial time if so is \( C \) using \( \mathcal{R} \).

3 Very Simple Grammars and Languages

Let \( G = (V_N, \Sigma, P, S) \) be a CFG in Greibach normal form. We say that \( G \) is in Greibach normal form in the strict sense if no right-hand side of the rule contains the starting nonterminal \( S \). It is well-known that every \( \lambda \)-free CFL is generated by a CFG in Greibach normal form in the strict sense([6]).

In what follows, all grammars we consider are assumed to be in Greibach normal form (not necessarily in the strict sense).

For each terminal symbol \( \alpha \in \Sigma \), a rule whose right-hand side is of the form \( \alpha a \) (where \( \alpha \in V_N \)) is called a-handle rule. Then, \( G \) is called very simple if for each \( \alpha \in \Sigma \), there exists exactly one a-handle rule in \( P \).

A language \( L \) is called very simple iff there exists a very simple CFG \( G \) such that \( L = L(G) \) holds. (Note that since every very simple grammar is \( \lambda \)-free, so is every very simple language.)

Example 1 Let \( \Sigma = \{a, b, c, d, e, f, g\} \). Consider a CFG \( G = (\{S, A, B, C, D\}, \Sigma, P, S) \), where \( P \) consists of the following:

\[
\begin{align*}
S & \rightarrow aABC, \quad A \rightarrow bAD \\
A & \rightarrow c, \quad B \rightarrow e \\
C & \rightarrow fC, \quad C \rightarrow g \\
D & \rightarrow d.
\end{align*}
\]

The grammar \( G \) is very simple and \( L(G) = \{a^{m}b^{m}c^{m}d^{m}e^{m}f^{m}g|m, n \geq 0\} \). Note that \( L(G) \) is non-regular. □

3.1 Characterization Results

The next result immediately follows from the definition.

Lemma 1 Let \( L \) be a very simple language. Then, for each string \( w \) in \( L \), if \( \text{lg}(w) \geq 2 \), then the initial symbol of \( w \) must differ from the last symbol of \( w \).

Example 2 The followings are not very simple languages: \( \{abba\} \), \( \{a^{n}|n \geq 1\} \), and \( \{c^{m}ac^{n}|m, n \geq 0\} \).

Thus, the class of very simple languages forms a proper subclass of simple deterministic languages by Korenjak and Hopcroft([7]) and is incomparable to the class of regular languages.

Lemma 2 For any very simple grammar \( G \), there exists a renaming \( f \) such that \( L(G) = f(D_{L}(G)) \), where \( D_{L}(G) \) is the derivation language of \( G \) under the left-most interpretation.

Since derivation languages under the left-most interpretation of CFGs of finite index (i.e., of non-terminal bounded CFGs) are regular([9]), it is proved that there exists a very simple language \( L \) which is of infinite index (i.e., not non-terminal bounded).

Lemma 3 For any very simple grammar in the strict sense \( G \), there exists a homomorphism \( h \) such that \( L(G) = h^{-1}(S_{D}) \), where \( D_{2} \) is a Dyck language over two letters, and \( S \) is a specific symbol neither in the alphabet for \( G \) nor \( D_{2} \).

Example 4 For any \( \lambda \)-free CFG \( G \) in Greibach normal form in the strict sense, there exist a coding \( f \) and a very simple grammar \( G' \) in the strict sense such that \( L(G) = f(L(G')) \).

Combining Lemmas 3 with 4, we have:

Theorem 5 For any \( \lambda \)-free CFL \( L \), there exist a coding \( f \), a homomorphism \( h \) such that \( L = fh^{-1}(SD_{2}) \).

This provides a simple, straightforward, alternative proof for this result which has been proved in [12], [13], [14].
3.2 Closure Properties

We can show that the class of very simple languages has none of standard closure property.

Theorem 6 The class of very simple languages is closed under none of the following: union, concatenation, intersection, complement, Kleene closure($+,*$), ($\lambda$-free) homomorphism, inverse homomorphism, intersection with regular languages, or reversal.

4 Identifying Very Simple Grammars

Let $L$ be a very simple language, where $L = L(G)$ for some very simple grammar $G = (V_N, \Sigma, P, S)$.

A rule of the form $A \rightarrow b$ is called terminal rule and a symbol $b$ is called terminating.

Lemma 7 Let $w, w_1, w_2$ be in $L$. Then, for each $a, b, c \in \Sigma$,

1. if $w = ax$ (for some $x \in \Sigma^*$), then the $a$-handle rule is of the form: $S \rightarrow ax$ (for some $a \in V_N$).

2. if $w = xa$ (for some $x \in \Sigma^*$), then the $a$-handle rule is of the form: $X_a \rightarrow a$ (for some $X_a \in V_N$).

3. if $\beta \Rightarrow^* a^n$ (for some $\beta \in V_N^n, n \geq 1$), then $\beta = X_a^n$, where $X_a$ is the left-hand side of the $a$-handle rule of the form: $X_a \rightarrow a$.

4. if $w_1 = x_1aby_1$ and $w_2 = x_2acy_2$ (where $x_i, y_i \in \Sigma^*$) and a symbol $a$ is not terminating, then the $b$-handle rule and the $c$-handle rule shares a common nonterminal $X$ as their left-hand sides, i.e., they are, respectively, of the forms: $X \rightarrow ba$ and $X \rightarrow cb$ (for some $a, b \in V_N$).

Proof. The proof is given for only 3, since others are all obvious. By induction on $n$. Let $\beta \Rightarrow^* a^m$.

From the property of very simple grammars, this holds if $X^{(i)} \rightarrow a$ is in $P$ and $\beta = X^{(i)}$. Suppose the claim holds for each $k < n$ and $\beta \Rightarrow^* a^n$. Let $\beta = X^{(i)}$ and $X \rightarrow ax$, then $\alpha^i \Rightarrow^* a^{n-1}$.

By the induction hypothesis $\alpha^i = X^{(i-1)}$, where $X^{(i-1)} \rightarrow a$ is in $P$. Further, since $X \rightarrow ax$ and $X_a \rightarrow a$, we have that $X = X_a$ and $\alpha = \lambda$. Thus, $\beta = X_a^n$ is obtained. □

4.1 Grammar Schema and Its Interpretations

Given a finite alphabet $\Sigma$, let $V_N = \{X_a | a \in \Sigma\} \cup \{S\}$ and let $PAR := \{x_a | a \in \Sigma\}$ be a finite set of parameters, where $S$ is a specific symbol not in $\{(X_a | a \in \Sigma) \cup \Sigma \cup PAR\}$, and the value of each parameter ranges among all elements from $V_N$. Let $\Gamma = (V_N \cup PAR)$. Then, a construct $X \rightarrow ax$, where $X \in V_N, a \in \Sigma$ and $x \in \Gamma^*$, is called rule form. We call a quadruplet $G = (V_N, \Sigma, P, S)$ grammar schema if $P$ is a finite set of rule forms.

An interpretation $I = (f_n, f_p)$ is an ordered pair of mappings, where $f_n$ is a coding defined on $V_N$, and $f_p$ is a homomorphism defined on $PAR$ such that for $\forall x \in PAR$, $f_p(x)$ is in $\Gamma^*$. Then, given a grammar schema $G$, let $I(G)$ be a quadruplet defined by $(f_n(V_N), \Sigma, I(P), S)$, where $I(P) = \{f_n(x) \rightarrow a f_p(x) | X \rightarrow ax \in P\}$. An interpretation $I$ is called ground if for $\forall x \in PAR$, $f_p(x) \in V_N$.

Let $G_0$ be the target grammar of inductive inference. We start with constructing the initial grammar schema $G_0 = (V_N, \Sigma, P_0, S)$, where $P_0 = \{X_a \rightarrow ax | a \in \Sigma\}$.

Our final goal here is to find(or identify) a ground interpretation $I = (f_n, f_p)$ such that $I(G_0)$ is equivalent to $G_4^\star$, i.e., $L(I(G_0)) = L(G_4)$.

In what follows, we claim that there effectively exists a finite set of positive data that ensures the identifiability of the target grammar. We proceed with our argument by using an example.

Let's consider the following very simple grammar $G_4 = (V_N, \Sigma, P, S)$ as the target grammar, where $V_N = \{S, A, B, C\}, \Sigma = \{a, b, c, d, e, f, g, h\}, P = \{S \rightarrow a A S, S \rightarrow c B, A \rightarrow b, B \rightarrow d B A, B \rightarrow e A, B \rightarrow f A C, C \rightarrow g C, C \rightarrow h\}$, and suppose that $R = \{aceb, cdeb, ceb, cfbh, cfghc, cfghc\} =$
\{w_1, \ldots, w_6\} is given as a sample set of \(L(G_\ast)\). We shall show that \(R\) is a sufficient sample set from which a ground interpretation \(I = (f_n, f_p)\) with the property that \(L(I(G_0)) = L(G_\ast)\) is obtained.

First, for \(w_1 = abceba\), since there are derivations: \(S \Rightarrow aX_b\gamma_1 = aX_b\gamma_1\) for some \(\gamma_1 \in \mathcal{V}_N^\ast\). Then, since \(X_b\gamma_1 \Rightarrow b\gamma_1 \Rightarrow \ast bceba\), \(\gamma_1\) must be \(X_b\gamma_1 \Rightarrow b\gamma_1 \Rightarrow \ast bceba\). Hence, \(\gamma_1 \Rightarrow \ast bceba\), and \(X_b\gamma_2 \Rightarrow \ast eb\).

Letting \(x_e\gamma_2 = X_e\gamma_3\), since \(X_e\gamma_3 \Rightarrow \ast x_e\gamma_3 \Rightarrow eb\), we have a set of relations

\[
x_a = X_b\gamma_1, \quad x_b\gamma_1 = X_c\gamma_2 \quad \text{where} \quad x_b = \lambda
\]

\[
x_c\gamma_2 = X_e\gamma_3, \quad x_e\gamma_3 = X_b \quad \text{from 3 of Lemma 7}.
\]

We generally say that a parameter \(x_a\) is empty if it is \(\lambda\). (Note that \(x_2\) is empty in our example.)

Applying the derivative computation to these leads to a constraint equation

\[
x_a = X_b(x_b), X_c(x_cX_e(x_e \setminus X_b))).
\]

By computing all empty parameters involved in the above equation, we have

\[
x_a(X_e \setminus x_cX_bX_c(X_a)) = X_b \cdots (c.w_1).
\]

In the same manner, from other five strings in \(R\), we have

\[
x_e(X_e \setminus x_dX_dX_c) = X_b \cdots (c.w_2)
\]

\[
x_e(X_e \setminus x_d) = X_b \cdots (c.w_3)
\]

\[
X_b \setminus x_fX_f \setminus x_c = X_h \cdots (c.w_4)
\]

\[
x_g(X_g \setminus x_fX_f \setminus x_c) = X_h \cdots (c.w_5)
\]

\[
x_g(X_g \setminus x_gX_g \setminus x_fX_f \setminus x_c)) = X_h \cdots (c.w_6).
\]

Let \(\text{Eq}(R) = \{(c.w_1), (c.w_2), \ldots, (c.w_6)\}\). Further, let

\[
\Sigma_s(R) = \{a \in \Sigma | \exists w \in R, \exists x \in \Sigma^\ast (w = ax)\}
\]

and

\[
\Sigma_f(R) = \{a \in \Sigma | \exists w \in R, \exists x \in \Sigma^\ast (w = xa)\}
\]

In our example, \(\Sigma_s(R) = \{a, c\}\) and \(\Sigma_f(R) = \{b, h\}\).

[Identifying Ground Interpretation \(I = (f_n, f_p)\)]

(1) Computation of \(f_n\): We assume \(\Sigma = \{a, b, c, d, e, f, g\}\) be an ordered set with this alphabetical order. An \(f_n\)-information is the information on identifying nonterminals obtained by specifying \(f_n\). We compute \(f_n\) from \(R\) as follows. There are three phases of computation of \(f_n\).

1. First, from 1 of Lemma 7, we may define \(f_n\) by

\[
\text{for } \forall a \in \Sigma_s(R), f_n(X_a) = S
\]

1. Note since \(c\) is in \(\Sigma_s(R)\), it is not terminating. Further, \("cd\", \("ce\" and \("cf\"\) are substrings in \(R\). Hence, from 4 of Lemma 7, we may define

\[
f_n(X_e) = f_n(X_f) = X_d.
\]

(This is also justified by simply observing the occurrences \((X_d \setminus x_c), (X_e \setminus x_c), (X_f \setminus x_c)\) in \(\text{Eq}(R)\), because these three occurrences imply that \(X_d, X_e\) and \(X_f\) must be identical as the prefix of \(x_c\).)

Further, from \((c.w_6)\), we have that \(x_g \in X_gX_h\). Hence, \(x_g \neq \lambda\), i.e., \(g\) is not terminating. Since \("gg\" and \("gh\"\) are substrings in \(R\), we may define

\[
f_n(X_h) = X_g.
\]

(2) Computation of \(f_p\): In order to obtain a concrete very simple grammar \(G\) which is at least consistent with the given data \(R\) (i.e., any string of \(R\) is generated by \(G\)), all we have to do is to solve a set of equations \(\text{Eq}(R)\) using \("f_n\)-information", so that we may obtain a ground interpretation \(I\) such that \(I(G_0)\) is consistent with \(R\). There are three phases of computing \(f_p\).

1. From 2 of Lemma 7, we may define \(f_p\) by

\[
\text{for } \forall a \in \Sigma_f(R), f_p(x_a) = \lambda
\]

That is, define \(f_p(x_b) = f_p(x_h) = \lambda\).

2. For each \(a \in \Sigma_f(R)\), we say that an equation \((c.w_i)\) is of type-\(a\) if its right-hand is of the form \(X_a^{k_i}\) for some positive integer \(k_i\). Then, the set \(\text{Eq}(R)\) is classified into \(\Sigma_f(R)\) blocks each of which consists of only equations of type-\(a\). In our example,

\[
b\text{-type block} = \{(c.w_1), (c.w_2), (c.w_6)\}
\]

and

\[
h\text{-type block} = \{(c.w_4), (c.w_5), (c.w_6)\}.
\]

For each \(a, b(\neq a) \in \Sigma_f(R)\), assume that \(X_a \neq X_b\). Let \(x_e\) be a parameter which appears in equations
of both type-\(a\) and type-\(b\). We say that \(x_c\) is double. Let \(x_c \backslash x_a\) and \(y \backslash x_c\) be its occurrences in equations of type-\(a\) and of type-\(b\), respectively, where \(x, y \in V^+_N\). Then, \(x_c = xX_a^k = yX_b^k(\exists k_1, k_2 \geq 0)\). Since \(X_a \neq X_b\), it holds that \(z_c = x = y\). Thus, double parameters can be always identified in this manner unless it is proved that \(f_n(X_b) = X_a\). In our example, since a parameter \(x_c\) is double and it is not proved that \(f_n(X_b) = X_b\), we have that \(X_a \backslash x_a = X_f \backslash x_c = \lambda\), i.e., \(f_p(x_c) = X_d\) (with a by-product: \(f_n(X_f) = X_d\) is obtained).

(2)-3. Constructing Associated Matrix: Let \(f_n(\text{Eq}(R)) = \{(e.w_i)|1 \leq vi \leq 6\}\), where \((e.w_i)\) is an equation obtained from \((e.w_i)\) by replacing each nonterminal \(X\) with \(f_n(X)\) and by removing double parameters already determined above:

\[
\begin{align*}
x_c(X_bS \backslash x_a) &= X_b \quad \cdots \quad (e.w_1) \\
x_c(X_a \backslash x_d) &= X_b^2 \quad \cdots \quad (e.w_2) \\
x_e &= X_b \quad \cdots \quad (e.w_3) \\
x_f &= X_b \quad \cdots \quad (e.w_4) \\
x_g &= X_bX_g \quad (x_f) = X_g \quad \cdots \quad (e.w_5) \\
x_g(X_bX_g) = (x_f) &= X_g \quad \cdots \quad (e.w_6).
\end{align*}
\]

Observing \(f_n(\text{Eq}(R))\), we note that an identical parameter \(x_f\) (or \(x_g\)) appears in two different constructs: for example, in \(X_b \backslash x_f, X_bX_g \backslash x_f\). From the derivational property of very simple grammars, \(x_f\) must have \(X_bX_g\) as its prefix. (Similarly, \(x_g\) has its prefix \(X_g\).

From these observation, let \(PAR' = \{x_a, x_d, x_e, x_f, x_g\}\) be an ordered set of distinct parameters obtained from \(PAR(= \{x_a|a \in \Sigma\})\) by removing all double parameters already determined and empty parameters. Then, let's consider a parameter replacement: \((y_a, y_d, y_e, y_f, y_g) = (X_bS \backslash x_a, X_d \backslash x_d, x_e, X_bX_g \backslash x_f, X_g \backslash x_g) \ldots \quad (*)\).

In \(f_n(\text{Eq}(R))\), for example, since \((e.w_6)\) is:

\[
x_gY_gY_gY_f = X_g,
\]

it may be taken as a formal linear relation:

\[
y_f + 2y_g = \lambda \quad \cdots \quad (f.w_6),
\]

where the operation “\(\dagger\)” is, in principle, a concatenation, but here we take it as “addition” defined by \(X_A^k \dagger X_i^k = X_b^{s+t} = (s+t)X_b^k\). Let's denote \((f.w_6)\) by:

\[
y_f + 2y_g = 0 \quad \cdots \quad (f.w_6).
\]

From \(f_n(\text{Eq}(R))\) we have a set of linear equations:

\[
\begin{align*}
y_a + y_d = 1 \cdots (\ell.w_1) & \quad y_f = 0 \cdots (\ell.w_4) \\
y_d + y_e = 2 \cdots (\ell.w_2) & \quad y_f + y_g = 0 \cdots (\ell.w_5) \\
y_e = 1 \cdots (\ell.w_3) & \quad y_f + 2y_g = 0 \cdots (\ell.w_6).
\end{align*}
\]

Then, construct a \((t \times m)\)-matrix \(M_R\) associated with \(R\) as follows:

\[
(i,j)-\text{entry of } M_R \overset{\text{def}}{=} \begin{cases}
\text{the coefficient of } j\text{-th parameter } y_{a_j}\text{ in } i\text{-th equation } (\ell.w_i),
\end{cases}
\]

where \(t = |R|, m = |PAR'| = \{|y_a, y_d, y_e, y_f, y_g\}|\), that is,

\[
M_R = \begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 0 \\
0 & 1 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 2 & 0
\end{pmatrix}
\]

And, we also have a matrix equation:

\[
M_RX^T = CT \quad \cdots \quad (E-1)
\]

where \(X = (y_a, y_d, y_e, y_f, y_g)\) is a solution-vector, \(C = (1,2,1,0,0,0)\) is a constant-vector obtained from the right-hand sides of all \((\ell.w_i)\).

Note since \(f_n(\text{Eq}(R))\) is now classified into \(s(= |\Sigma_f(R)|)\) disjoint blocks, \(PAR''\) is also a disjoint union of \(s\) blocks corresponding to small matrices, that is, \(M_R\) is of the form:

\[
M_R = \begin{pmatrix}
M_1 & 0 & 0 & 0 \\
0 & M_2 & 0 & 0 \\
0 & 0 & \cdots & 0 \\
0 & 0 & 0 & M_s
\end{pmatrix},
\]

where \(M_i\) corresponds to equations of type-\(b_i(b_i \in \Sigma_f(R))\).

Thus, in order to obtain a ground interpretation \(I = (f_n, f_p)\) such that \(I(y_0)\) is consistent with \(R\), all we have to do is to solve an equation \((E-1)\) in a usual manner in linear algebra. That is,

\[
\begin{pmatrix}
1 & 0 & 1 & 0 & 0 & 1 \\
0 & 1 & 1 & 0 & 0 & 2 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 2 & 0
\end{pmatrix} \xrightarrow{\ast} \begin{pmatrix}
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 1 & 1 & 0 \\
0 & 0 & 0 & 1 & 2 & 0
\end{pmatrix}
\]
Thus, we have a solution:

\[ X = (y_{a}, y_{d}, y_{e}, y_{f}, y_{g}, \mathbf{y}) = (0, 1, 0, 0, 0, 0), \text{ and} \]
\[ (x_{a}, x_{d}, x_{e}, x_{f}, x_{g}) = (X_{b}S, X_{d}X_{b}, X_{b}, X_{b}X_{g}, X_{g}). \]

Hence, the solution makes unique interpretation \( I = (f_{n}, f_{p}) \), where

\[
\begin{align*}
 f_{n}(X_{a}) &= S \\
 f_{n}(X_{c}) &= S \\
 f_{n}(X_{e}) &= X_{d} \\
 f_{n}(X_{f}) &= X_{d} \\
 f_{n}(X_{h}) &= X_{g}
\end{align*}
\]
\[
\begin{align*}
 f_{p}(x_{a}) &= X_{b}S \\
 f_{p}(x_{c}) &= \lambda \\
 f_{p}(x_{f}) &= X_{b}X_{g} \\
 f_{p}(x_{g}) &= X_{g} \\
 f_{p}(x_{h}) &= \lambda.
\end{align*}
\]

The rule set of \( I(G_{0}) \) consists of

\[
\begin{align*}
 S &\rightarrow aX_{b}S, \quad X_{a} \rightarrow b, \quad S \rightarrow cX_{d}, \\
 X_{d} &\rightarrow dX_{d}X_{b}, \quad X_{d} \rightarrow eX_{b}, \quad X_{d} \rightarrow fX_{b}X_{g}, \\
 X_{g} &\rightarrow gX_{g}, \quad X_{g} \rightarrow h.
\end{align*}
\]

It is easily seen that \( I(G_{0}) \) is equivalent (in fact, isomorphic) to \( G_{*} \). (Note that a grammar \( I(G_{0}) \) we have in the manner described above is not always isomorphic to, but equivalent to \( G_{*} \), which will be proved later.)

In some case, we may have indeterminate parameters in the solution of (E-1). A famous fact in linear algebra tells us:

**Lemma 8** (i) The equation (E-1) has a solution iff the rank of \( M_{R} \) is equal to that of \( M_{R}C^{T} \). (ii) Let \( m = t \) in (E-1). Then, \( M_{R} \) is non-singular iff (E-1) has the unique solution.

Thus, if \( m = \text{rank}(M_{R}) \), then the equation (E-1) has the unique solution. And, if \( m > t \) or \( m \neq \text{rank}(M_{R}) \), then (E-1) is solved in part. That is, let \( \text{Sol}(Eq(R)) \) be the set of solutions of \( Eq(R) \). Then, we have that \( \text{Sol}(Eq(R)) = P-\text{Sol}(Eq(R)) \cup \text{Uns}(Eq(R)) \), where \( P-\text{Sol}(Eq(R)) \) is the set of solutions partly solved, and \( \text{Uns}(Eq(R)) \) is the set of equations involved in indeterminate parameters left unsolved.

Then, by assigning an appropriate value to each indeterminate parameter in \( \text{Uns}(Eq(R)) \), it is always possible to have a complete solution for \( f_{n}(Eq(R)) \), which completes the definition of \( f_{p} \), i.e., \( f_{p}(x_{a}) = u_{a}\beta_{a} \), where \( u_{a} \) is a string in \( V_{N}^{*} \) satisfying \( y_{a} = u_{a}x_{a} \) in (E-1), and \( \beta_{a} \) is the value for \( y_{a} \) in the solution \( X \). (Actually it is recognized that (\( t \), \( u_{a} \)) is redundant for computing (E-1) because (\( t \), \( w_{0} \)) = (2(\( t \), \( w_{0} \)) + (1)(\( t \), \( w_{1} \)), and that rank(\( M_{R} \) = rank(\( M_{R}C^{T} \)) = 5, while \( M_{R} \) (where \( R \) = \( R - \{w_{6}\} \)) is non-singular. That is, \( Eq(R) \) (or \( Eq(R) \)) is completely solved.

Returning to our example, instead of \( R \), let's suppose that we are given a sample set \( R' = \{w_{1}, w_{2}, w_{3}, w_{5}\} \). Then, we eventually have:

\[ \text{Sol}(Eq(R')) = \{x_{a} = X_{b}S, x_{d} = X_{d}X_{b}, x_{e} = X_{b}\} \cup \{y_{f} + y_{g} = 1\}. \]

For instance, a solution where \( y_{f} = 0 \) and \( y_{g} = 1 \) leads to \( I' = (f_{n}', f_{p}') \), where

\[
\begin{align*}
 f_{n}'(X_{a}) &= S \\
 f_{n}'(X_{c}) &= S \\
 f_{n}'(X_{e}) &= X_{d} \\
 f_{n}'(X_{f}) &= X_{d} \\
 f_{n}'(X_{h}) &= X_{g}
\end{align*}
\]
\[
\begin{align*}
 f_{p}'(x_{a}) &= X_{b}S \\
 f_{p}'(x_{c}) &= \lambda \\
 f_{p}'(x_{f}) &= X_{b}X_{g} \\
 f_{p}'(x_{g}) &= X_{g} \\
 f_{p}'(x_{h}) &= \lambda.
\end{align*}
\]

which eventually provides a grammar \( I'(G_{0}) \) not equivalent to \( G_{*} \).

Thus, in any case, we have a ground interpretation \( I = (f_{n}, f_{p}) \) obtained from \( Eq(R) \) by solving equations. We say that \( I \) is an instance of \( Eq(R) \).

### 4.2 Characteristic Samples

Let \( G = (V_{N}, \Sigma, P, S) \) be any very simple grammar. A finite subset \( R \) of \( L(G) \) is called characteristic sample of \( G \) if \( L(G) \) is the smallest very simple language containing \( R \).

Given a very simple grammar \( G \) with a terminal alphabet \( \Sigma \), let \( R \) be a finite subset of \( L(G) \). Then, we say that \( Eq(R) \) is linearly dependent iff so is \( \{\text{vec}(w)\mid w \in R\} \), where \( \text{vec}(w) \) is a vector \((c_{1}, \ldots, c_{m})\), each \( c_{j} \) is a coefficient of \( y_{j} \) in the left-hand side of a linear equation (\( t \), \( w \)) in \( f_{n}(Eq(R)) \). \( Eq(R) \) is linearly independent iff it is not linearly dependent. Further, we say that
(c.w) is a linear combination of $\text{Eq}(R)$ iff the corresponding row vector $\text{vec}(w)$ is a linear combination of $\text{Vec}(R) = \{\text{vec}(w)|w \in R\}$.

Let $I_R = \{I|I = (f_n, f_p)\}$ be an instance of $\text{Eq}(R)$. Further, let $I = (f_n', f_p')$ and $I' = (f_n'', f_p'')$ be in $I_R$ and $I_{R'}$, respectively, where $R' \subseteq R \subseteq L(G)$ and $\text{alph}(R') = \text{alph}(R') = \Sigma$. Then, $I$ is called a refinement of $I'$ if there is a coding $f: V_N \rightarrow V_N$ such that for all $a \in \Sigma$, $f(f_n'(x_a)) = f_n(x_a)$ and $f_p(f_p'(x_a)) = f_p(x_a)$, where $G_0 = (V_N, \Sigma, P_0, S)$. (Note that $I$ is a refinement of $I'$ implies that $L(I'(G_0)) \subseteq L(I(G_0))$. See, e.g., $I$ and $I'$ discussed in the previous example.)

Let $G = (V_N, \Sigma, P, S)$ be a very simple grammar. For each $a \in \Sigma$, let $u_a$ be the first shortest string in $\Sigma^*$ in the lexicographic order such that $S \Rightarrow^* u_a\alpha a_{\alpha a}$ for some $\alpha_a \in V_N^*$ in $G$. Further, by short($\alpha_a$) we denote the set of all the shortest strings in $\Sigma^*$ derivable from $\alpha_a$.

Let $R_G$ be defined by

$$R_G = \bigcup_{\alpha \in \Sigma} \{u_{a\alpha} v_a \in L(G) | v_a \in \text{short}(\alpha_a)\}.$$  

$R_G$ is called representative sample of $G$. (Note that $u_{a\alpha} v_a = u_{b\alpha} v_b (a \neq b)$ may occur.) We can show that $R_G$ is a sufficient set of positive data from which a very simple grammar equivalent to $G$ is identified.

**Lemma 9** For all $I \in I_{R_G}$, it holds that $L(I(G_0)) = L(G)$.

**Lemma 10** Let $R$ be a finite subset of $L(G)$, where $G = (V_N, \Sigma, P, S)$ be a very simple grammar. Then, $R$ is a characteristic sample of $G$ iff it holds that for all $I \in I_R$, $L(I(G_0)) = L(G)$.

From Lemma 10, we immediately have:

**Lemma 11** Let $R$ be a characteristic sample of a very simple grammar $G$ and let $R'$ be a finite set such that $R \subseteq R' \subseteq L(G)$. Then, it holds that for all $I \in I_{R'}$, $L(I(G_0)) = L(G)$.

**Corollary 12** For any very simple grammar $G$, the representative sample $R_G$ of $G$ is a characteristic sample of $G$.

### 4.3 Identification Algorithm and Its Time Complexity

Let $G_* = (V_N, \Sigma, P, S)$ be the target grammar. We now present an identification algorithm $IA$ which is consistent, responsive and conservative ([1]). In Figure 1, $G_{\Sigma}$ denotes $(S) \cup V_N, \Sigma, P, \Sigma, S)$, where $V_N, \Sigma = \{X_a | a \in \Sigma\}, P_\Sigma = \{X_a \Rightarrow ax_a | a \in \Sigma\}$. We shall show that $IA$ given in Figure 1 below eventually identifies in the limit a grammar $G_R$ such that $L(G_*) = L(G_R)$, where $R$ is the set of positive data provided.

From Lemma 11 and Corollary 12, it follows:

**Lemma 13** Let $G_{R_0}, G_{R_1}, ..., G_{R_i}, ...$ be the sequence of conjectured grammars produced by $IA$, where $G_{R_i} = I_i(G_{0})$. Then, (i) for all $i \geq 0$, $R_i \subseteq L(G_{R_i})$, and (ii) there exists $r \geq 0$ such that for all $i \geq 0$, $L(G_{R_i}) = L(G_{R_{i+r}}) = L(G_*)$.

Thus, we have the following:

**Theorem 14** Given any very simple grammar $G_*$, the algorithm $IA$ identifies in the limit a very simple grammar $G_R$ such that $L(G_*) = L(G_R)$, where $R$ is the set of positive data provided.

Since the cardinality of a maximal linearly independent subset of $R_G$ is bounded by $|\Sigma|$, in the repeat loop the number of times when an input string $w_i$ is not in $L(G_{R_{i+1}})$ is bounded by $|\Sigma|$. That is, the number of implicit errors of prediction is bounded by $|\Sigma|$. It takes at most $O(m^3)$ time to compute $\text{Eq}(R_i)$, which comes from that it is reduced to the computation of an inverse matrix with $m$-dimension, where $m = \text{Max}_{w_j \in R_i}(|\Sigma|, \text{lg}(w_j))$.

**Notes.** (1) Computing $\text{Eq}(R_i)$ actually requires time less than $O(m^3)$, because it is possible to gain time efficiency greatly by making use of partial solutions of $P_{-\text{Sol}(\text{Eq}(R_{i-1}))}$ and reducing the dimension of matrix $M_{R_i} C_i^T$ considerably. (2) The size of $G_*$ (denoted by $|G_*|$) may be defined by $|V_N| + |P|$. Then, since $|V_N| \leq |\Sigma|$ and $|P| = |\Sigma|$, it holds that $|\Sigma| \leq |G_*| \leq 2|\Sigma|$.

Thus, we have:

**Theorem 15** The algorithm $IA$ requires at most $O(m^3)$ time for updating a conjecture at each
step, and the number of implicit errors of prediction is bounded by $|\Sigma|$, where $\Sigma$ is the terminal alphabet of the target grammar $G_*$, and $m = \max_{w_j \in R_i} \{|\Sigma|, \lg(w_j)|\}$, $R_i$ is the set of data provided up to step $i$.

Since the polynomial-time identifiability in the limit implies the polynomial-time identifiability via equivalence queries ([10]) and the latter implies the polynomial-time PAC-identifiability ([4]), we have:

Corollary 16 The class of very simple grammars is identifiable in polynomial time via only equivalence queries, where only positive counterexamples are required in the identification process. Further, it is also PAC-identifiable in polynomial time.

Input: a positive presentation of a very simple language $L(G_*)$
Output: a sequence of very simple grammars

Procedure
- initialize $R_0 = \emptyset$;
- initialize the grammar schema $G_{0,\emptyset}$;
- let $G_{R_0} = (\{S\}, \emptyset, \emptyset, S)$;
- let $i = 1$;
- repeat (forever)
  - read the next positive example $w_i$;
  - let $R_i = R_{i-1} \cup \{w_i\}$;
  - let $\text{alph}(R_i) = \text{alph}(R_{i-1}) \cup \text{alph}(w_i)$;
  - if $w_i \in L(G_{R_{i-1}})$, then
    - let $G_{R_i} = G_{R_{i-1}}$;
    - output $G_{R_i}$;
  - else
    - augment $G_{0,\Sigma}$ using $\Sigma = \text{alph}(R_i)$;
    - compute a constraint equation $(c.w_i)$ for $w_i$;
    - let $\text{Eq}(R_i) = \text{Eq}(R_{i-1}) \cup \{(c.w_i)\}$;
    - compute $\text{Sol}(\text{Eq}(R_i))$ by solving $\mathcal{M}_{R_i}A_t = C_t$;
    - make an instance $I_1$ of $\text{Eq}(R_i)$ from $\text{Sol}(\text{Eq}(R_i))$;
    - output $G_{R_i} = I_1(G_{0,\Sigma})$;

Figure 1. The Identification Algorithm $IA$

5 Conclusions

(1) We have shown that the class of very simple grammars is identifiable in the limit from positive data, and presented an algorithm which identifies any very simple grammar in polynomial time in the sense of Pitt.

In the identifiability criteria discussed by Pitt, the class of regular languages (or even the class of zero-reversible languages) is not identifiable in the limit in polynomial time using DFAs ([10]). Recently, the author shows that the subclass of regular languages called strictly $k$-testable languages is identifiable in the limit in polynomial time from positive data using DFAs ([15]), which is, to author's knowledge, the first positive result ever obtained concerning the polynomial time identifiability in the sense of Pitt.

Quite recently, Mäkinen ([8]) discusses the problem of learning Szilard languages of linear grammars and gives a linear-time algorithm for solving the problem. From the definition, the class of Szilard languages of linear grammars is clearly a proper subclass of the class of very simple languages, and is also properly included in the class of zero-reversible languages ([2]). Further, the class of very simple languages is incomparable to the class of zero-reversible languages. (See Figure 2.)

(2) Roughly, one of the recent results by Shinohara ([11]) shows that the class of $\lambda$-free grammars having a given fixed number of rules is identifiable in the limit from positive data, which implies the identifiability of the class of very simple grammars with a fixed size of terminal alphabet. It is, however, to be noted that the algorithm given in this paper allows us to identify any grammar with arbitrary size of the terminal alphabet. In other words, the algorithm works for the class of very simple languages over the growing alphabet. More significantly, the Shinohara's result implies only the identifiability in the limit from positive data, and it does not tell or even suggest any non-enumerative, efficient ("polynomial-time" in the sense of Pitt) algorithm for this class.
Figure 2. Language Classes Relations

References


[5] P. Butzbach. Une famille de congruences de thue pour lesquelles le probleme de l'équivalence est decidable. application a l'équivalenc des grammaires sepa-


