数理解析研究所講究録
第 754 巻 1991 年 15-24

Very Simple Grammars and Polynomial-Time Learning*

Takashi YOKOMORI
横森 貴

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-

* Supported in part by Grants-in-Aid for Scientific Research No.02650261 from the Ministry of Education, Science and Culture, Japan.

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 neatest 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 any 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 for identifying $\mathcal{C}$ in the limit using $\mathcal{R}$ 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, and for any presentation of L, the number of implicit errors of prediction made by A is at most $p(n)$, and the time used by A between receiving the i-th example w_i and outputting the i-th conjectured representation r_i is at most $q(n, m_1 + \cdots + m_i)$, where $m_j = lg(w_j)$.

Finally, we say that $\mathcal{R}$ is *identifiable in the limit in polynomial time* if so is $\mathcal{C}$ 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 λ-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 $a \in \Sigma$, a rule whose right-hand side is of the form $a\alpha$(where $\alpha \in V_N^*$) is called *a-handle rule*. Then, G is called *very simple* if for each a in Σ, 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 λ-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 :

$$\begin{array}{ll} S \to aABC, & A \to bAD \\ A \to c, & B \to e \\ C \to fC, & C \to g \\ D \to d. & \end{array}$$

The grammar G is very simple and $L(G) = \{ab^m cd^m ef^n g \mid m, n \geq 0\}$. Note that $L(G)$ is nonregular. □

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 $lg(w) \geq 2$, then the initial symbol of w must differ from the last symbol of w.*

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

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 nonterminal bounded *CFGs*) are regular([9]), it is proved that there exists a very simple language L which is of infinite index(i.e., not nonterminal bounded).

Lemma 3 *For any very simple grammar in the strict sense G, there exists a homomorphism h such that $L(G) = h^{-1}(\$D_2)$, where D_2 is a Dyck language over two letters, and \$ is a specific symbol neither in the alphabet for G nor D_2.*

Lemma 4 *For any λ-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 λ-free CFL L, there exist a coding f, a homomorphism h such that $L = fh^{-1}(\$D_2)$.*

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(+, *), (λ-free) homomorphim, inverse homorphism, 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 $A \to b$ 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 Σ^*), then the a-handle rule is of the form : $S \to a\alpha$ (for some $\alpha \in V_N^*$).*
2. *if $w{=}xa$ (for some x in Σ^*), then the a-handle rule is of the form : $X_a \to a$ (for some $X_a \in V_N$).*
3. *if $\beta \Longrightarrow^* a^n$ (for some $\beta \in V_N^+, n \geq 1$), then $\beta = X_a^n$, where X_a is the left-hand side of the a-handle rule of the form : $X_a \to a$).*
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 : $X \to b\alpha$ and $X \to c\beta$ (for some $\alpha, \beta \in V_N^*$).*

Proof. The proof is given for only 3, since others are all obvious. By induction on n. Let $\beta \Longrightarrow^* a$. From the property of very simple grammars, this holds iff $X_a \to a$ is in P and $\beta = X_a$. Suppose the claim holds for each $k < n$ and $\beta \Longrightarrow^* a^n$. Let $\beta = X\beta'$ and $X \to a\alpha$, then $\alpha\beta' \Longrightarrow^* a^{n-1}$. By the induction hypothesis $\alpha\beta' = X_a^{n-1}$, where $X_a \to a$ is in P. Further, since $X \to a\alpha$ and $X_a \to a$, 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 Σ, 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 $X \to ax$, where $X \in V_N$, $a \in \Sigma$ and $x \in \Gamma^*$, is called *rule form.* We call a quadruple $\mathcal{G} = (V_N, \Sigma, P, S)$ *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 Γ^*. Then, given a grammar schema $\mathcal{G}$, let $I(\mathcal{G})$ be a quadruple defined by $(f_n(V_N), \Sigma, I(P), S)$, where $I(P) = \{f_n(X) \to af_p(x) | X \to ax \in P\}$. An interpretaion I is called *ground* if for $\forall x \in PAR$, $f_p(x) \in V_N^*$.

Let G_* be the target grammar of inductive inference. We start with constructing the initial grammar schema $\mathcal{G}_0 = (V_N, \Sigma, P_0, S)$, where $P_0 = \{X_a \to ax_a | a \in \Sigma\}$.

Our final goal here is to find(or *identify*) a ground interpretation $I = (f_n, f_p)$ such that $I(\mathcal{G}_0)$ is equivalent to G_*, i.e., $L(I(\mathcal{G}_0)) = L(G_*)$.

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 $G_* = (V_N, \Sigma, P, S)$ as the target grammar, where $V_N = \{S, A, B, C\}, \Sigma = \{a, b, c, d, e, f, g, h\}, P = \{S \to aAS, S \to cB, A \to b, B \to dBA, B \to eA, B \to fAC, C \to gC, C \to h\}$, and suppose that $R = \{abceb, cdebb, ceb, cfbh, cfbgh, cfbggh\} =$

$\{w_1, ..., w_6\}$ is given as a sample set of $L(G_*)$. We shall show that R is a sufficient sample set from which a ground interpretation $I = (f_n, f_p)$ with the property that $L(I(\mathcal{G}_0)) = L(G_*)$ is obtained.

First, for $w_1 = abceb$, since there are derivations: $S \Longrightarrow ax_a$ and $x_a \Longrightarrow^* bceb$, x_a must be $X_b\gamma_1$ for some $\gamma_1 \in V_N^*$. Then, since $x_a \Longrightarrow b\gamma_1 \Longrightarrow^* bceb$, γ_1 must be $X_c\gamma_2$ for some $\gamma_2 \in V_N^*$. Hence, $\gamma_1 \Longrightarrow cx_c\gamma_2 \Longrightarrow^* ceb$, and $x_c\gamma_2 \Longrightarrow^* eb$. Letting $x_c\gamma_2 = X_e\gamma_3$, since $X_e\gamma_3 \Longrightarrow ex_e\gamma_3 \Longrightarrow eb$, we have a set of relations

$$\begin{array}{ll} x_a = X_b\gamma_1, & x_b\gamma_1 = X_c\gamma_2 \quad (\text{where } x_b = \lambda) \\ x_c\gamma_2 = X_e\gamma_3, & x_e\gamma_3 = X_b \quad (\text{from 3 of Lemma 7}). \end{array}$$

We generally say that a parameter x_a is *empty* if it is λ. (Note that x_b is empty in our example.)

Applying the *derivative computation* to these leads to a constraint equation

$$x_a = X_b(x_b\backslash X_c(x_c\backslash X_e(x_e\backslash X_b))).$$

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

$$x_e(X_e\backslash x_c(X_bX_c\backslash x_a)) = X_b \quad \cdots \quad (c.w_1).$$

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

$$\begin{aligned} x_e(X_e\backslash x_d(X_d\backslash x_c)) &= X_b^2 \cdots (c.w_2) \\ x_e(X_e\backslash x_c) &= X_b \cdots (c.w_3) \\ X_b\backslash x_f(X_f\backslash x_c) &= X_h \cdots (c.w_4) \\ x_g(X_bX_g\backslash x_f(X_f\backslash x_c)) &= X_h \cdots (c.w_5) \\ x_g(X_g\backslash x_g(X_bX_g\backslash x_f(X_f\backslash x_c))) &= X_h \cdots (c.w_6). \end{aligned}$$

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

$$\begin{aligned} \Sigma_s(R) &= \{a \in \Sigma \mid \exists w \in R, \exists x \in \Sigma^*(w = ax)\} \\ &\text{and} \\ \Sigma_f(R) &= \{a \in \Sigma \mid \exists w \in R, \exists x \in \Sigma^*(w = xa)\}. \end{aligned}$$

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)-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)-2. 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 $(X_d\backslash x_c), (X_e\backslash x_c), (X_f\backslash x_c)$ in Eq(R_1), 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

$$f_n(X_h) = X_g.$$

(This is also obtained by observing the occurrence $x_g(X_g\backslash x_g) \in \{X_h\}$ in $(c.w_6)$.)

(1)-3. Further, for other $X (\in \{X_b, X_d, X_g\})$ not defined yet, we define $f_n(X) = X$.

(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 Eq(R) using "f_n-information", so that we may obtain a ground interpretation I such that $I(\mathcal{G}_0)$ is consistent with R. There are three phases of computing f_p.

(2)-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 $\quad f_p(x_b) = f_p(x_h) = \lambda.$

(2)-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 Eq(R) is classified into $|\Sigma_f(R)|$ blocks each of which consists of only equations of type-a. In our example,

$$\begin{aligned} b\text{-type block} &= \{(c.w_1), (c.w_2), (c.w_3)\} \quad \text{and} \\ h\text{-type block} &= \{(c.w_4), (c.w_5), (c.w_6)\}. \end{aligned}$$

For each $a, b (\neq a) \in \Sigma_f(R)$, assume that $X_a \neq X_b$. Let x_c be a parameter which appears in equations of both type-a and type-b. We say that x_c is *double*. Let $x\backslash x_c$ 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, $x_c = xX_a^{k_1} = yX_b^{k_2} (\exists k_1, k_2 \geq 0)$. Since $X_a \neq X_b$, it holds that $x_c = x = y$. 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 $f_n(X_h) = X_b$, we have that $X_d\backslash x_c = X_f\backslash x_c = \lambda$, 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 $f_n(\mathrm{Eq}(R)) = \{(e.w_i)) | 1 \leq \forall i \leq 6\}$, where $(e.w_i)$ is an equation obtained from $(c.w_i)$ by replacing each nonterminal X with $f_n(X)$ and by removing double parameters already determined above :

$$\begin{aligned}
x_e(X_bS\backslash x_a) &= X_b \quad \cdots \quad (e.w_1)\\
x_e(X_d\backslash x_d) &= X_b^2 \quad \cdots \quad (e.w_2)\\
x_e &= X_b \quad \cdots \quad (e.w_3)\\
X_b\backslash x_f &= X_g \quad \cdots \quad (e.w_4)\\
x_g(X_bX_g\backslash x_f) &= X_g \quad \cdots \quad (e.w_5)\\
x_g(X_g\backslash x_g(X_bX_g\backslash x_f)) &= X_g \quad \cdots \quad (e.w_6).
\end{aligned}$$

Observing $f_n(\mathrm{Eq}(R))$, 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: $(y_a, y_d, y_e, y_f, y_g) = (X_bS\backslash x_a, X_d\backslash x_d, x_e, X_bX_g\backslash x_f, X_g\backslash x_g) \quad \cdots \quad (*)$. In $f_n(\mathrm{Eq}(R))$, for example, since $(e.w_6)$ is: $X_g y_g y_g y_f = X_g$, it may be taken as a formal linear relation :

$$y_f \dot{+} 2y_g \;=\; \lambda \quad \cdots \quad (f.w_6),$$

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

$$y_f + 2y_g \;=\; 0 \quad \cdots \quad (\ell.w_6).$$

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

$$\begin{array}{ll}
y_a + y_e = 1 \cdots (\ell.w_1) & y_f = 0 \cdots (\ell.w_4)\\
y_d + y_e = 2 \cdots (\ell.w_2) & y_f + y_g = 0 \cdots (\ell.w_5)\\
y_e = 1 \cdots (\ell.w_3) & y_f + 2y_g = 0 \cdots (\ell.w_6).
\end{array}$$

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

(i,j)-entry of $M_R \overset{\mathrm{def}}{\Longleftrightarrow}$
"the coefficient of j-th parameter y_{a_j} in i-th equation $(\ell.w_i)$",

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

$$M_R = \begin{pmatrix}
1 & 0 & 1 & 0 & 0\\
0 & 1 & 1 & 0 & 0\\
0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 0\\
0 & 0 & 0 & 1 & 1\\
0 & 0 & 0 & 1 & 2
\end{pmatrix}.$$

And, we also have a matrix equation:

$$M_R \mathcal{X}^T = \mathcal{C}^T \quad \cdots \quad \text{(E-1)}$$

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

Note since $f_n(\mathrm{Eq}(R))$ 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:

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

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 $I(\mathcal{G}_0)$ 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,

$$\left(\begin{array}{ccccc|c}
1 & 0 & 1 & 0 & 0 & 1\\
0 & 1 & 1 & 0 & 0 & 2\\
0 & 0 & 1 & 0 & 0 & 1\\
0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 0 & 1 & 2 & 0
\end{array}\right) \Rightarrow^* \left(\begin{array}{ccccc|c}
1 & 0 & 0 & 0 & 0 & 0\\
0 & 1 & 0 & 0 & 0 & 1\\
0 & 0 & 1 & 0 & 0 & 1\\
0 & 0 & 0 & 1 & 0 & 0\\
0 & 0 & 0 & 1 & 1 & 0\\
0 & 0 & 0 & 1 & 2 & 0
\end{array}\right)$$

$$\Rightarrow^* \left(\begin{array}{ccccc|c} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{array}\right).$$

Thus, we have a solution :

$$\mathcal{X} = (y_a, y_d, y_e, y_f, y_g) = (0,1,1,0,0), \text{ and}$$
$$(x_a, x_d, x_e, x_f, x_g) = (X_bS, X_dX_b, X_b, X_bX_g, X_g).$$

Hence, the solution makes unique interpretation $I = (f_n, f_p)$, where

$$\begin{array}{lcl} f_n(X_a) & = & S \\ f_n(X_c) & = & S \\ f_n(X_e) & = & X_d \\ f_n(X_f) & = & X_d \\ f_n(X_h) & = & X_g \end{array} \qquad \begin{array}{lcl} f_p(x_a) & = & X_bS \\ f_p(x_b) & = & \lambda \\ f_p(x_c) & = & X_d \\ f_p(x_d) & = & X_dX_b \end{array}$$
$$\begin{array}{lcl} f_p(x_e) & = & X_b \\ f_p(x_f) & = & X_bX_g \\ f_p(x_g) & = & X_g \\ f_p(x_h) & = & \lambda. \end{array}$$

The rule set of $I(\mathcal{G}_0)$ consists of

$$\begin{array}{lll} S \to aX_bS, & X_b \to b, & S \to cX_d, \\ X_d \to dX_dX_b, & X_d \to eX_b, & X_d \to fX_bX_g, \\ X_g \to gX_g, & X_g \to h. & \end{array}$$

It is easily seen that $I(\mathcal{G}_0)$ is equivalent (in fact, isomorphic) to G_*. (Note that a grammar $I(\mathcal{G}_0)$ 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_RC^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 Sol(Eq(R)) be the set of solutions of Eq(R). Then, we have that Sol(Eq(R)) = P-Sol(Eq(R)) $\cup$ Uns(Eq(R)), where P-Sol(Eq(R)) is the set of solutions partly solved, and 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 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 $y_a = u_a\backslash x_a$ in (*), and β_a is the value for y_a in the solution $\mathcal{X}$. (Actually it is recognized that $(\ell.w_6)$ is redundant for computing (E-1) because $(\ell.w_6) = 2(\ell.w_5)+(-1)(\ell.w_4)$, and that $\text{rank}(M_R) = \text{rank}(M_RC^T) = 5$, while $M_{\tilde{R}}$ (where $\tilde{R} = R - \{w_6\}$) is non-singular. That is, Eq(R)(or Eq($\tilde{R}$)) 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 : Sol(Eq(R')) $= \{x_a = X_bS, x_d = X_dX_b, x_e = X_b\} \cup \{y_f + y_g = 1\}$.

For instance, a solution where $y_f = 0$ and $y_g = 1$ leads to $I' = (f'_n, f'_p)$, where

$$\begin{array}{lcl} f'_n(X_a) & = & S \\ f'_n(X_c) & = & S \\ f'_n(X_e) & = & X_d \\ f'_n(X_f) & = & X_d \end{array} \qquad \begin{array}{lcl} f'_p(x_a) & = & X_bS \\ f'_p(x_b) & = & \lambda \\ f'_p(x_c) & = & X_d \\ f'_p(x_d) & = & X_dX_b \end{array}$$
$$\begin{array}{lcl} f'_p(x_e) & = & X_b \\ f'_p(x_f) & = & X_bX_g \\ f'_p(x_g) & = & X_h \\ f'_p(x_h) & = & \lambda. \end{array}$$

which eventually provides a grammar $I'(\mathcal{G}_0)$ 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 Σ, let R be a finite subset of $L(G)$. Then, we say that Eq(R) is *linearly dependent* iff so is $\{\text{vec}(w) | w \in R\}$, where vec($w$) is a vector $(c_1, ..., c_m)$, each c_j is a coefficient of y_j in the left-hand side of a linear equation $(\ell.w)$ 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* Eq(R) iff the corresponding row vector vec(w) is a linear combination of Vec$(R) = \{\text{vec}(w) | w \in R\}$.

Let $\mathcal{I}_R = \{I | I = (f_n, f_p)$ is an instance of Eq$(R)\}$. Further, let $I = (f_n, f_p)$ and $I' = (f'_n, f'_p)$ be in $\mathcal{I}_R$ and $\mathcal{I}_{R'}$, respectively, where $R' \subseteq R \subset L(G)$ and alph(R)=alph(R')=Σ. Then, I is called a *refinement of* I' if there is a coding $f : V_N \to V_N$ such that for $\forall a \in \Sigma$, $f(f'_n(X_a)) = f_n(X_a)$ and $f_n(f'_p(x_a)) = f_p(x_a)$, where $\mathcal{G}_0 = (V_N, \Sigma, P_0, S)$. (Note that I is a refinement of I' implies that $L(I'(\mathcal{G}_0)) \subseteq L(I(\mathcal{G}_0))$. 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 Σ^* in the lexicographic order such that $S \Rightarrow^* u_a a \alpha_a$ for some $\alpha_a \in V_N^*$ in G. Further, by short(α_a) we denote the set of all the shortest strings in Σ^* derivable from α_a.

Let R_G be defined by

$$R_G = \bigcup_{a \in \Sigma} \{u_a a v_a \in L(G) | v_a \in \text{short}(\alpha_a), \; u_a \text{ and short}(\alpha_a) \text{ are defined above}\}.$$

R_G is called *representative sample* of G. (Note that $u_a a v_a = u_b b v_b (a \neq b)$ 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* $I \in \mathcal{I}_{R_G}$, *it holds that* $L(I(\mathcal{G}_0)) = L(G)$.

Lemma 10 *Let R be a finite subset of $L(G)$, where $G = (V_N, \Sigma, P, S)$ a very simple grammar. Then, R is a characteristic sample of G iff it holds that for all $I \in \mathcal{I}_R$, $L(I(\mathcal{G}_0)) = L(G)$.*

From Lemma 10, we immediately have:

Lemma 11 *Let R be a charaterisitc sample of a very simple grammar G and let R' be a finite set such that $R \subset R' \subset L(G)$. Then, it holds that for all $I \in \mathcal{I}_{R'}$, $L(I(\mathcal{G}_0)) = L(G)$.*

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, $\mathcal{G}_{0,\Sigma}$ denotes $(\{S\} \cup V_{N,\Sigma}, \Sigma, P_\Sigma, S)$, where $V_{N,\Sigma} = \{X_a | a \in \Sigma\}$, $P_\Sigma = \{X_a \to a x_a | a \in \Sigma\}$. 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 $G_{R_i} = I_i(\mathcal{G}_0)$. Then, (i) for $\forall i \geq 0$, $R_i \subseteq L(G_{R_i})$, and (ii) there exists $r \geq 0$ such that for $\forall i \geq 0$, $L(G_{R_r}) = L(G_{R_{r+i}}) = L(G_*)$.*

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 Eq(R_i), which comes from that it is reduced to the computation of an inverse matrix with m-dimension, where $m = \text{Max}_{w_j \in R_i}\{|\Sigma|, lg(w_j)\}$.

Notes. (1) Computing Eq(R_i) actually requires time less than $O(m^3)$, becase it is possible to gain time efficiency greatly by making use of partial solutions of P-Sol(Eq(R_{i-1})) and reducing the dimension of matrix $\mathcal{M}_{R_i} C_i^T$ 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 Σ is the terminal alphabet of the target grammar G_*, and $m = \mathrm{Max}_{w_j \in R_i}\{|\Sigma|, lg(w_j)\}$, 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 $\mathcal{G}_{0,\emptyset}$;
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 alph(R_i)= alph(R_{i-1}) $\cup$ {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 $\mathcal{G}_{0,\Sigma}$ using Σ =alph(R_i);
compute a constraint equation (c.w_i) for w_i;
let Eq(R_i)=Eq(R_{i-1}) $\cup$ {($c.w_i$)} ;
compute Sol(Eq(R_i)) by solving $\mathcal{M}_{R_i}\mathcal{X}_i^T = \mathcal{C}_i^T$;
make an instance I_i of Eq(R_i) from Sol(Eq(R_i));
output $G_{R_i} = I_i(\mathcal{G}_{0,\Sigma})$;

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 *DFA*s([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 *DFA*s([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 λ-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.

24

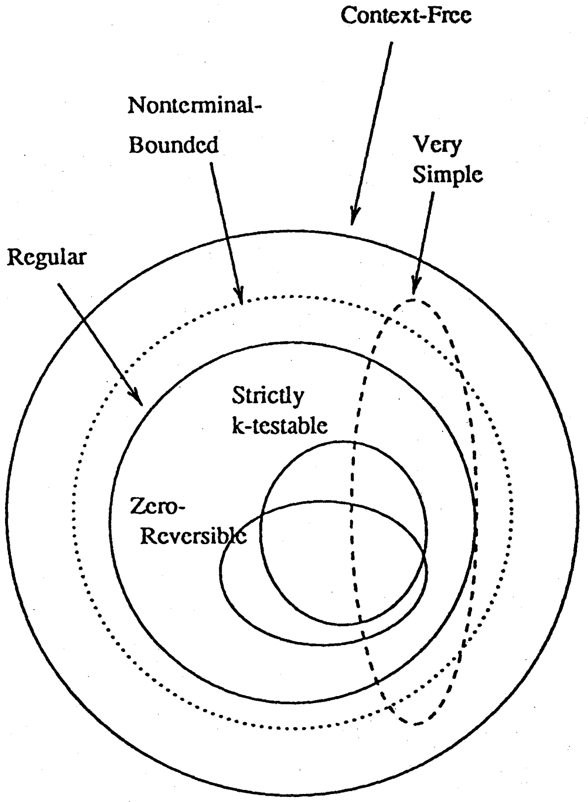

Figure 2. Language Classes Relations

References

[1] D. Angluin. Inductive inference of formal languages from positive data. *Information and Control*, 45:117–135, 1980.

[2] D. Angluin. Inference of reversible languages. *Journal of the ACM*, 29:741–765, 1982.

[3] D. Angluin. Learning regular sets from queries and counter-examples. *Information and Computation*, 75:87–106, 1987.

[4] D. Angluin. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[5] P. Butzbach. Une famille de congruences de thue pour lesquelles le probleme de l'equivalence est decidable. application a l'equivalenc des grammaires separees. In M. Nivat, editor, *Automata, Languages and Programming*, pages 3–12. North-Holland/American Elsevier, 1973.

[6] M. A. Harrison. *Introduction to Formal Language Theory*. Addison-Wesley, Reading, Massachusetts, 1978.

[7] A. Korenjak and J. Hopcroft. Simple deterministic languages. In *Proceedings of 7th Annual IEEE Conference on Switching and Automata Theory*, pages 36–46, 1966.

[8] E. Makinen. The grammatical inference problem for the Szilard languages of linear grammars. *Information Processing Letters*, 36:203–206, 1990.

[9] E. Moriya. The associate language and the derivation complexity of formal grammars. *Information and Control*, 22:139–162, 1973.

[10] L. Pitt. Inductive inference, DFAs, and computational complexity. In *Proceedings of 2st Workshop on Analogical and Inductive Inference*, Lecture Notes in Artificial Intelligence, Springer-Verlag 397, pages 18–44, 1989.

[11] T. Shinohara. Inductive inference from positive data is powerful. In *Proceedings of 3rd Workshop on Computational Learning Theory*, pages 97–110, 1990.

[12] P. Turakainen. Characterizations of simple transducers and principal semi AFLs in terms of morphisms and inverse morphisms. *J. of Inform. Process. Cybern. EIK*, 23:403–413, 1987.

[13] P. Turakainen. On characterization of language families in terms of inverse morphisms. *International Journal of Computer Mathematics*, 16:189–200, 1987.

[14] T. Yokomori. On purely morphic characterizations of contex-free languages. *Theoretical Comput. Sci.*, 51:301–308, 1987.

[15] T. Yokomori. A note on the polynomial-time identification of strictly local languages in the limit. Report CSIM 90-03, Dept. of Comupt. Sci. and Inf. Math., Univ. of Elect.-Communi., 1990.